

LAPORAN TUGAS BESAR

IF2110 Algoritma dan Struktur Data



Purry Mail


Dipersiapkan oleh:

K01 - B

Shannon Aurellius Anastasya Lie (13523019)
Muhammad Raihan Nazhim Oktana (13523021)
Joel Hotlan Haris Siahaan (13523025)
Najwa Kahani Fatima (13523043)
Muhammad Iqbal Haidar (13523111)

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		IF2110-TB-01-B		36
		Revisi	0	13-12-2024

STEI- ITB	IF2110 LaporanTB K01B	Halaman 1 dari 36 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.		

Daftar Isi

1	Ringkasan	3
2	Penjelasan Tambahan Spesifikasi Tugas	4
2.1	Starred Email	4
3	Struktur Data (ADT)	4
3.1	ADT Datetime (ADT Sederhana)	4
3.2	ADT List dengan Struktur Data Array Statik	5
3.3	ADT List dengan Struktur Data Array Dinamik	6
3.4	ADT Matriks	7
3.5	ADT Mesin Karakter dan Mesin Kata	8
3.6	ADT Stack	9
3.7	ADT Priority Queue (ADT Queue)	9
3.8	ADT Tree	11
3.9	ADT Inbox	12
3.10	ADT List Email	13
3.11	ADT List User	14
3.12	ADT Pengguna	15
4	Program Utama	16
5	Algoritma-Algoritma Menarik	17
5.1	N Ary Tree dalam Representasi Binary Tree	17
6	Data Test	18
6.1	Insialisasi	18
6.2	Pengguna	18
6.3	Draft Email	20
6.4	Inbox Utama Email	22
6.5	Balas Email	25
6.6	Status Kepentingan	26
6.7	Notifikasi	27
6.8	Save & Load	27
6.9	Starred Email	29
7	Test Script	30
8	Pembagian Kerja dalam Kelompok	31
9	Lampiran	32
9.1	Deskripsi Tugas Besar 1	32
9.2	Notulen Rapat	33
9.3	Log Activity Anggota Kelompok	33

1 Ringkasan

Tugas Besar Mata Kuliah IF2110 - Algoritma dan Struktur Data Tahun Akademik 2024/2025 adalah berupa pembuatan program email yang memanfaatkan berbagai struktur data. Tugas besar ini dilatarbelakangi oleh terjadinya krisis keamanan di organisasi OWCA akibat kerentanan pada aplikasi klien surel mereka (ZMail). Kerentanan tersebut dimanfaatkan oleh pihak luar untuk meretas sistem dan mencuri informasi rahasia. Merespons masalah tersebut, Mr. Monogram memerintahkan pengembangan aplikasi klien surel pengganti bernama **PurryMail**. Namun, karena Purry tidak bisa memprogram kode, mahasiswa diberi tugas untuk membantu Purry untuk menciptakan aplikasi PurryMail yang berbasis struktur data ini.

Dengan menyelesaikan Tugas Besar Algoritma dan Struktur Data, mahasiswa diharapkan mampu mengenali dan memahami konsep dan dasar struktur data. Selain itu, mahasiswa diharapkan mampu untuk mendesain dan mengimplementasikan struktur data pada program sesuai dengan kebutuhan. Mahasiswa juga diharapkan mampu melakukan pemecahan masalah menggunakan struktur data yang sesuai. Lalu, mahasiswa diharapkan mampu menggunakan struktur data yang telah disediakan oleh pustaka.

Aplikasi PurryMail yang dibuat harus memenuhi spesifikasi dan kriteria berikut:

1. Berbasis CLI (Command-Line Interface), menggunakan bahasa pemrograman C.
2. Memanfaatkan struktur data (ADT) yang telah dipelajari di mata kuliah.
3. Program harus dapat dikompilasi dan dijalankan pada sistem operasi berbasis UNIX, menggunakan Makefile untuk automasi kompilasi.
4. Program wajib memanfaatkan berbagai ADT, dengan penjelasan alasan penggunaan setiap ADT untuk masing-masing fitur.
5. Setiap ADT harus diuji melalui unit-testing dengan driver yang sesuai.

Terdapat beberapa ADT yang wajib digunakan yaitu ADT Datetime, ADT List dengan Struktur Data Array Statik, ADT List dengan Struktur Data Array Dinamik, ADT Matriks Mesin Karakter dan Mesin Kata, ADT Stack, ADT Priority Queue (ADT Queue), dan ADT Tree. Terdapat juga beberapa spesifikasi utama yang ada pada program ini termasuk inisialisasi, perintah, pengguna, draft email, inbox utama email, balas email, status kepentingan, notifikasi, save, load, dan konfigurasi program.

Laporan ini berisi penjelasan tambahan spesifikasi tugas, deskripsi struktur data yang digunakan, program utama, algoritma-algoritma menarik yang digunakan, data test, test script, pembagian kerja kelompok, dan lampiran. Laporan ini juga menjelaskan keluaran yang didapat

dari tugas ini, yaitu suatu program email dengan pemanfaatan berbagai jenis struktur data, baik yang sudah ada dalam pustaka, maupun yang dimodifikasi sesuai kebutuhan.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 Starred Email

Untuk mengimplementasikan fitur ini, diperlukan spesifikasi tambahan pada file `email.config` yakni tambahan data starred. Juga pada fitur INBOX akan ada kolom tambahan untuk menunjukkan apakah email tersebut starred atau tidak.

Fitur ini memiliki tiga jenis perintah yang dapat dimasukkan oleh pengguna yakni;

a. STAR <EMAILXXX>

Fitur ini digunakan untuk menandai email terkait menjadi starred. Apabila pengguna memasukkan email yang sudah starred maka program akan memberikan pesan informasi bahwa email sudah di starred.

b. UNSTAR <EMAILXXX>

Fitur ini digunakan untuk menghapus tanda starred pada email terkait. Apabila pengguna memasukkan email yang sudah di unstarred maka program akan memberikan pesan informasi bahwa email sudah di unstarred.

c. DAFTAR_STARRED

Fitur ini digunakan untuk menampilkan inbox dengan email yang di starred saja.

3 Struktur Data (ADT)

3.1 ADT Datetime (ADT Sederhana)

ADT Datetime memiliki peran sebagai representasi struktur data yang menyimpan waktu. ADT ini digunakan pada implementasi fitur Draft Email dan Notifikasi sebab pada fitur ini terdapat timestamp yang berkaitan dengan waktu. Terdapat 1 tipe data bentukan dan 8 fungsi pada ADT ini yang masing-masing dijelaskan spesifikasinya pada file header `datetime.h`. Tipe data bentukan dan beberapa fungsi utama pada ADT ini antara lain,

Tipe Data Bentukan

1. Datetime

```
typedef struct {  
    int day;  
    int month;
```

```

int year;
int hour;
int minute;
int second;
int epoch;
} Datetime;

```

Tipe data ini menyimpan data waktu yakni; int day, int month, int year, int hour, int minute, int second, int epoch. Data-data ini yang digunakan untuk implementasi fitur-fitur pada program utama.

Fungsi/Prosedur

1. void CreateDatetime(Datetime *t);
Prosedur ini digunakan untuk inisialisasi variabel dengan tipe data bentukan Datetime. F.S. adalah menghasilkan sebuah t kosong, dimana semua komponen t bernilai NIL.
2. void getDatetime(Datetime *t);
Prosedur ini digunakan untuk mengambil waktu lokal komputer dan menyimpannya pada variabel dengan tipe data bentukan Datetime. F.S. adalah komponen t berisi data waktu lokal komputer terkini.
3. int toEpoch(const char *timestamp);
Prosedur ini digunakan untuk mengkonversi string timestamp dengan format YYYY-MM-DD HH:MM:SS menjadi total detik sejak Unix Epoch. Fungsi ini mengembalikan detik dalam int.

3.2 ADT List dengan Struktur Data Array Statik

ADT List Statik memiliki peran sebagai representasi struktur data List dengan alokasi memori secara statik. ADT ini tidak secara langsung digunakan pada fitur, namun konsep list statik ini diaplikasikan pada ADT List User yang menyimpan data user pada program. Alasan penggunaan list secara statis karena program membatasi jumlah maksimal user yang dapat ditampung sehingga cukup diperlukan list secara statik saja. Terdapat 1 tipe data bentukan dan 21 fungsi pada ADT ini yang masing-masing dijelaskan spesifikasinya pada file header *liststatik.h*. Tipe data bentukan dan beberapa fungsi utama pada ADT ini antara lain,

Tipe Data Bentukan

1. ListStatik

```

#define CAPACITY 100
#define MARK -9999

```

```
typedef int ElType;
typedef int IdxType;
typedef struct {
    ElType contents[CAPACITY];
} ListStatik;
```

Tipe data ini digunakan sebagai memori tempat menyimpan (container) elemen dengan tipe data `ElType` sebanyak `CAPACITY` dari `List`.

Fungsi/Prosedur

1. `void CreateListStatik(ListStatik *l);`
Prosedur ini digunakan untuk inisialisasi variabel dengan tipe data bentukan `ListStatik`. F.S. Terbentuk `List` 1 kosong dengan kapasitas `CAPACITY` dan elemen bernilai `MARK`.
2. `void insertAt(ListStatik *l, ElType val, IdxType idx);`
Prosedur ini digunakan untuk menambahkan `val` sebagai elemen pada index `idx` di `List`. F.S. `val` adalah elemen yang disisipkan pada index `idx` 1.
3. `void deleteAt(ListStatik *l, ElType *val, IdxType idx);`
Prosedur ini digunakan untuk menghapus elemen pada index `idx` `List`. F.S. `val` menyimpan nilai elemen pada index `idx` 1 sebelum penghapusan dan panjang `List` berkurang satu.

3.3 ADT List dengan Struktur Data Array Dinamik

ADT List Dinamik memiliki peran sebagai representasi struktur data `List` dengan alokasi memori secara dinamis. ADT ini tidak secara langsung digunakan pada fitur, namun konsep list dinamik ini diaplikasikan pada ADT List Email yang menyimpan data email pada program. Alasan penggunaan list secara dinamis karena program tidak membatasi jumlah email yang dapat ditampung sehingga diperlukan list secara dinamis. Terdapat 1 tipe data bentukan dan 24 fungsi pada ADT ini yang masing-masing dijelaskan spesifikasinya pada file header *listdin.h*. Tipe data bentukan dan beberapa fungsi utama pada ADT ini antara lain,

Tipe Data Bentukan

1. `ListDin`

```
typedef char ElType;
typedef int IdxType;
typedef struct
```

```

{
    ElType *buffer;
    int nEff;
    int capacity;
} ListDin;

```

Tipe data ini digunakan sebagai memori tempat menyimpan (container) elemen dengan tipe data `ElType` dari `List`, selain itu juga terdapat `int nEff` untuk menyimpan banyaknya elemen efektif dan `int capacity` yang menyimpan kapasitas dari `List`.

Fungsi/Prosedur

1. `void CreateListDin(ListDin *l, int capacity);`
Prosedur ini digunakan untuk inisialisasi variabel dengan tipe data bentukan `ListDin`. F.S. Terbentuk `List l` kosong `nEff = 0` dengan kapasitas `capacity`.
2. `void expandList(ListDin *l, int num);`
Prosedur ini digunakan untuk menambahkan `capacity` sebanyak `num` pada `List`. F.S. Ukuran `List` bertambah sebanyak `num`.
3. `void shrinkList(ListDin *l, int num);`
Prosedur ini digunakan untuk mengurangi `capacity` sebanyak `num` pada `List`. F.S. Ukuran `List` berkurang sebanyak `num`.

3.4 ADT Matriks

ADT Matriks memiliki peran sebagai representasi struktur data `Matriks` dengan alokasi memori secara statis. ADT ini digunakan pada implementasi fitur Status Kepentingan sebab pada fitur ini diperlukan hubungan data tiap user dengan user lainnya. Terdapat 1 tipe data bentukan dan 26 fungsi pada ADT ini yang masing-masing dijelaskan spesifikasinya pada file header *matrix.h*. Tipe data bentukan dan beberapa fungsi utama pada ADT ini antara lain,

Tipe Data Bentukan

1. `Matrix`

```

#define ROW_CAP 100
#define COL_CAP 100

typedef int IdxType;
typedef struct {
    int mem[ROW_CAP][COL_CAP];
}

```

```

    int rowEff;
    int colEff;
} Matrix;

```

Tipe data ini digunakan sebagai memori tempat menyimpan (container) elemen dengan tipe data `ElType` sebanyak `ROW_CAP × COL_CAP` dari `Matrix`, selain itu juga terdapat `int rowEff` dan `int colEff` untuk menyimpan banyaknya baris serta kolom efektif dari `Matrix`.

Fungsi/Prosedur

1. `void createMatrix(int nRows, int nCols, Matrix *m);`
Prosedur ini membentuk sebuah `Matrix` "kosong" yang siap diisi berukuran `nRow x nCol` di "ujung kiri" memori. F.S. Matriks `m` terdefinisi kosong.
2. `boolean isIdxEffMatrix(Matrix m, IdxType i, IdxType j);`
Fungsi ini mengirimkan `true` jika `i, j` adalah index yang valid pada `Matrix m`.
3. `void copyMatrix(Matrix mIn, Matrix *mOut);`
Prosedur ini menyalin semua elemen pada `Matrix mIn` menuju `Matrix mOut`.

3.5 ADT Mesin Karakter dan Mesin Kata

ADT Mesin Karakter dan Mesin Kata memiliki peran sebagai representasi sebuah mesin yang mengolah karakter maupun kata pada program. ADT ini digunakan pada implementasi fitur Perintah, Load & Save sebab fitur-fitur ini menerima dan memanipulasi input string dari pengguna ataupun file. Terdapat 1 tipe data bentukan dan 13 fungsi pada ADT ini yang masing-masing dijelaskan spesifikasinya pada file header *mesinkarakter.h* & *mesinkata.h*. Tipe data bentukan dan beberapa fungsi utama pada ADT ini antara lain,

Tipe Data Bentukan

1. `Word`

```

#define NMax 1000

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

```


Tipe data ini digunakan sebagai memori tempat menyimpan (container) elemen/karakter yang membangun sebuah kata dengan tipe data `char` sebanyak `NMax`. Selain itu, juga terdapat `int Length` yang menyimpan panjang kata/banyaknya karakter pada `Word`.

Fungsi/Prosedur

1. `void STARTWORD();`
Prosedur ini memulai membaca kata dan kemudian mengukusisinya kedalam sebuah variable bertipe `Word`.
2. `void ADVWORD();`
Prosedur ini membaca kata selanjutnya dan kemudian mengukusisinya kedalam sebuah variable bertipe `Word`.
3. `void CopyWord();`
Prosedur ini merupakan proses mengukusisi sebuah kata dengan menyalin tiap karakter kedalam sebuah variable bertipe `Word`.

3.6 ADT Stack

ADT Stack memiliki peran sebagai representasi sebuah struktur data Stack. ADT ini digunakan pada implementasi fitur Draft Email sebab pada fitur ini terdapat perintah UNDO, REDO yang keduanya dapat diimplementasikan dengan push/pop pada Stack sehingga isi draft dapat kembali ke versi sebelumnya. Terdapat 2 tipe data bentukan dan 5 fungsi pada ADT ini yang masing-masing dijelaskan spesifikasinya pada file header *stack.h*. Tipe data bentukan dan beberapa fungsi utama pada ADT ini antara lain,

Tipe Data Bentukan

1. Stack

```
#define Nil -1
#define MaxEl 100

typedef emailType infotype;
typedef int address;
typedef struct {
    infotype T[MaxEl];
    address TOP;
} Stack;
```

Tipe data bentukan ini digunakan sebagai memori tempat menyimpan (container) elemen dengan tipe data `InfoType` dari `Stack`, selain itu juga terdapat `address TOP` untuk menyimpan alamat elemen teratas dari `Stack`.

Fungsi/Prosedur

1. `void CreateEmptyStack(Stack *S);`
Prosedur ini digunakan untuk menginisialisasi variable bertipe `Stack` yakni, membuat sebuah `Stack S` yang kosong berkapasitas `MaxEl`. F.S. `S` terdefinisi, `TOP` bernilai `NIL`.
2. `void PushStack(Stack * S, infotype X);`
Prosedur ini digunakan untuk menambahkan `X` sebagai elemen `Stack S`. F.S. `X` menjadi `TOP` yang baru, `TOP` bertambah satu.
3. `void PopStack(Stack * S, infotype* X);`
Prosedur ini digunakan untuk menghapus `X` dari `Stack S`. F.S. `X` adalah nilai elemen `TOP` yang lama, `TOP` berkurang satu.

3.7 ADT Priority Queue (ADT Queue)

ADT Priority Queue memiliki peran sebagai representasi sebuah struktur data Queue. ADT ini digunakan pada implementasi fitur Notifikasi sebab pada fitur tersebut diperlukan urutan menampilkan email berdasarkan waktu terkirimnya secara menaik. Terdapat 2 tipe data bentukan dan 8 fungsi pada ADT ini yang masing-masing dijelaskan spesifikasinya pada file header *prioqueue.h*. Tipe data bentukan dan beberapa fungsi utama pada ADT ini antara lain,

Tipe Data Bentukan

1. `Notif`

```
typedef struct notif {  
    int timeEpoch;  
    char *timestamp;  
    char *subject;  
} Notif;
```

Tipe data bentukan ini digunakan untuk menyimpan data-data email yang penting untuk implementasi fungsi Notifikasi antara lain; `int timeEpoch`, `char timestamp`, dan `char subject`. `timeEpoch` menyimpan jumlah detik `timestamp` sejak Epoch Unix sedangkan `timestamp` dan `subject` merupakan komponen dari email. `Notif` ini merupakan tipe data elemen pada Queue.

2. `PrioQueue`

```
#define CAPACITYPRIOQUEUE 100
```

```
typedef Notif ElTypePrioQ;
typedef struct {
    ElTypePrioQ buffer[CAPACITYPRIOQUEUE];
    int idxHead;
    int idxTail;
} PrioQueue;
```

Tipe data bentukan ini digunakan sebagai memori tempat menyimpan (container) elemen dengan tipe data `ElTypePrioQ` (Notif) dari Queue. Selain itu, juga terdapat `int idxHead` dan `int idxTail` yang menyimpan index elemen pertama dan terakhir dari Queue.

Fungsi/Prosedur

1. `void CreatePrioQueue(PrioQueue *pq);`
Prosedur ini membentuk sebuah Priority Queue `pq` yang kosong, yakni kondisi `idxHead` dan `idxTail` bernilai `IDX_UNDEF`.
2. `void enqueueAscending(PrioQueue *pq, ElTypePrioQ val);`
Prosedur ini menambahkan `val` pada `pq` dengan aturan FIFO. F.S. `val` disisipkan pada posisi yang sesuai, `idxTail` "mundur" dalam buffer melingkar. `pq` terurut membesar.
3. `void dequeue(PrioQueue *pq, ElTypePrioQ *val);`
Prosedur ini menghapus `val` pada `pq` dengan aturan FIFO. F.S. `val` berisi nilai elemen `idxHead` sebelum penghapusan, `idxHead` "mundur".

3.8 ADT Tree

ADT Matriks memiliki peran sebagai representasi struktur data `BinaryTree` dengan alokasi memori secara dinamis. ADT ini digunakan pada implementasi fitur Balas Pesan dan Baca Pesan. Terdapat 2 tipe data bentukan dan 22 fungsi pada ADT ini yang masing-masing dijelaskan spesifikasinya pada file header *tree.h*. Tipe data bentukan dan beberapa fungsi utama pada ADT ini antara lain :

Tipe Data Bentukan

1. `TreeNode`

```
typedef int ElTypeTree;
typedef struct treeNode {
    ElTypeTree info;
    Address left;
    Address right;
```

```
} TreeNode;
```

Tipe data ini digunakan sebagai memori tempat menyimpan (container) elemen info dengan tipe data `ElType` dari `Tree`, selain itu juga terdapat `left` dan `right` untuk menyimpan alamat *child* sebelah kiri dan kanan.

2. Address

```
typedef struct treeNode* Address;  
typedef Address Tree;
```

Tipe data ini merupakan alias untuk pointer ke tipe data `TreeNode`. Penamaan ini digunakan untuk mempermudah pemahaman dan penulisan kode ke dalam program.

Fungsi/Prosedur

1. `Address newNode(ElTypeTree val)`
Fungsi `newTreeNode` mengalokasikan alamat di memori sebesar `TreeNode` dan menyimpan nilai `val` (`ElTypeTree`) yang diterima ke info dari node. Fungsi ini mengembalikan `Address` yaitu pointer ke `TreeNode` yang dibuat.
2. `void addReply(Tree *p, ElTypeTree new_reply)`
Prosedur `addReply` mengalokasikan alamat di memori sebesar `TreeNode` dan menyimpan nilai `val` (`ElTypeTree`) yang diterima ke info dari node. Prosedur ini menetapkan node tersebut sebagai anak dari pohon `p`.
3. `void removeNode(Tree *p, Address node)`
Prosedur `removeNode` menghapus node dengan alamat `node` pada pohon `p` dengan mendealokasi memorinya.

3.9 ADT Inbox

Dalam ADT Inbox terdapat delapan fungsi yang dideklarasikan dalam sebuah file header “*inbox.h*”. File tersebut mencakup fungsi-fungsi yang diperlukan untuk mengimplementasikan fitur inbox pada aplikasi berbasis CLI. Setiap fungsi memiliki perannya masing-masing dalam memastikan inbox dapat berfungsi secara efektif. Berikut merupakan perincian untuk setiap fungsi yang ada :

1. Fungsi `ceil_division` melakukan pembagian dua bilangan bulat `a` (`integer`) dan `b` (`integer`) dengan hasil pembulatan ke atas (`ceiling`). Fungsi ini mengembalikan hasil pembagian `a / b` yang dibulatkan ke atas. Fungsi ini digunakan menentukan jumlah halaman untuk daftar email pada inbox.

2. Fungsi `formatEmailID` mengubah sebuah ID email numerik menjadi format string tertentu, misalnya `EMAIL001`. Fungsi ini menerima dua parameter: `emailID (int)` yaitu ID numerik email yang akan diformat dan `output (char*)` yaitu merupakan buffer untuk menyimpan hasil string yang diformat. Fungsi ini digunakan untuk memastikan setiap email memiliki ID dalam format yang konsisten untuk kemudahan identifikasi dan penampilan.
3. Fungsi `truncateString` memotong string input menjadi panjang tertentu, dengan hasil disimpan di string output. Fungsi ini menerima tiga parameter yaitu input (`const char*`) yang merupakan String asli yang akan dipotong, output (`char*`) yang merupakan buffer untuk menyimpan string yang telah dipotong, dan `maxLength (int)` merupakan panjang maksimum string hasil potongan. Fungsi ini digunakan untuk menghindari tampilan string yang terlalu panjang di CLI, misalnya untuk subjek atau isi email yang hanya perlu ditampilkan sebagian.
4. Fungsi `DisplayInbox` menampilkan daftar email yang terdapat dalam struktur data `ListEmail` ke layar. Fungsi ini menerima parameter `listEmail (ListEmail)` yang merupakan struktur data yang berisi daftar email. Fungsi ini digunakan untuk menampilkan informasi email seperti pengirim, subjek, dan tanggal secara terorganisasi dalam CLI.
5. Fungsi `bacaPesanInbox` menampilkan isi pesan dari email tertentu dalam inbox berdasarkan `emailID`. Fungsi ini menerima dua parameter yaitu `listEmail (ListEmail)` merupakan struktur data yang berisi daftar email dan `emailID (int)` merupakan ID email yang ingin dibaca. Fungsi ini memungkinkan pengguna untuk membaca isi email spesifik dari daftar inbox.
6. Fungsi `bacaPesan` membaca isi pesan email berdasarkan `emailID` setelah validasi berhasil dilakukan. Fungsi ini menerima parameter `emailID (int)` yang merupakan ID email yang valid untuk dibaca. Fungsi ini berfungsi sebagai langkah terakhir untuk membaca pesan setelah ID email divalidasi, misalnya memastikan bahwa email tersebut ada dalam daftar.
7. Fungsi `Inbox` merupakan fungsi utama yang mengelola, menjalankan semua operasi terkait inbox, dan mengatur keseluruhan proses manajemen inbox, termasuk menampilkan daftar email, membaca pesan, memvalidasi email ID, dan navigasi.
8. Fungsi `starEmailInbox` digunakan untuk menandai sebuah email sebagai "starred". Mencari email berdasarkan `emailID`. Jika ditemukan, memeriksa apakah email sudah di-starred atau belum. Jika belum di-starred, status email diubah menjadi starred dan pesan keberhasilan ditampilkan.
9. Fungsi `starEmail` adalah wrapper untuk memanggil fungsi `starEmailInbox` dengan parameter `listEmail` dan `emailID`.
10. Fungsi `unStarEmailInbox` digunakan untuk menghapus tanda "starred" dari sebuah email. Mencari email berdasarkan `emailID`. Jika ditemukan, memeriksa apakah email sudah di-starred. Jika sudah di-starred, status diubah menjadi tidak starred, dan pesan keberhasilan ditampilkan.
11. Fungsi `unStarEmail` adalah wrapper untuk memanggil fungsi `unStarEmailInbox` dengan parameter `listEmail` dan `emailID`.

12. Fungsi `countEmailStarredInbox` ini menghitung jumlah email pada inbox yang bertanda "starred" untuk pengguna tertentu. Memeriksa email berdasarkan ID penerima (`idPenerima`) dan email yang dicopy (`idCC`). Mengembalikan jumlah email yang memenuhi kriteria.
13. Fungsi `StarredInbox` menampilkan daftar email yang bertanda "starred" dalam inbox. Jika tidak ada email yang starred, menampilkan pesan bahwa daftar kosong. Jika ada, menampilkan informasi email seperti Email ID, Subject, Pengirim, Status, dan Timestamp dalam format tabel. Mendukung pagination untuk menampilkan email secara bertahap jika jumlahnya banyak.

3.10 ADT List Email

ADT List Email memiliki peran sebagai representasi sebuah struktur data yang menyimpan informasi terkait dengan Email. ADT ini digunakan pada implementasi seluruh fitur pada program, sebab ADT ini menyimpan data-data email yang penting untuk nantinya dimanipulasi di setiap fitur pada program. Terdapat 2 tipe data bentukan dan 8 fungsi pada ADT ini yang masing-masing dijelaskan spesifikasinya pada file header *listemail.h*. Tipe data bentukan dan beberapa fungsi utama pada ADT ini antara lain,

Tipe Data Bentukan

1. `emailType`

```
typedef struct {  
    int id;  
    int idPengirim;  
    int idPenerima;  
    int idCC;  
    char* timestamp;  
    char* subyek;  
    char* body;  
    int reply;  
    boolean read;  
    boolean readCC;  
    boolean starred;  
} emailType ;
```

Tipe data bentukan ini digunakan untuk menyimpan seluruh data terkait sebuah email antara lain; `int id`, `int idPengirim`, `int idPenerima`, `int idCC`, `char timestamp`, `char subyek`, `char body`, `int reply`, `boolean read`, `boolean readCC`, dan `boolean starred`. Semua data diatas merupakan komponen pembentuk sebuah email sesuai dengan spesifikasi yang telah diberikan.

2. ListEmail

```
typedef int IdxType;
typedef struct
{
    emailType *data;
    int number;
    int capacity;
} ListEmail;
```

Tipe data bentukan ini digunakan sebagai memori tempat menyimpan (container) elemen dengan tipe data emailType dari ListEmail, selain itu juga terdapat int number untuk menyimpan banyaknya seluruh email pada program dan int capacity sebagai kapasitas maksimum ListEmail untuk menampung email.

Fungsi/Prosedur

1. void CreateListEmail(ListEmail *l, int capacity);
Prosedur ini digunakan untuk inisialisasi membentuk ListEmail l kosong dengan kapasitas sebesar capacity.
2. void addEmail(ListEmail *l, emailType email);
Prosedur ini digunakan untuk menambah elemen email ke dalam ListEmail l.
3. void expandListEmail(ListEmail *l, int num);
Prosedur ini digunakan untuk memperbesar kapasitas dari ListEmail l sebanyak num. F.S. capacity bertambah sebanyak num.

3.11 ADT List User

ADT List User memiliki peran sebagai representasi sebuah struktur data yang menyimpan informasi terkait dengan User. ADT ini digunakan pada implementasi seluruh fitur pada program, sebab ADT ini menyimpan data-data user yang penting untuk nantinya dimanipulasi di setiap fitur pada program. Terdapat 2 tipe data bentukan dan 6 fungsi pada ADT ini yang masing-masing dijelaskan spesifikasinya pada file header *listuser.h*. Tipe data bentukan dan beberapa fungsi utama pada ADT ini antara lain,

Tipe Data Bentukan

1. userType

```
typedef struct {
    int id;
```

```

char* email;
char* password;
} userType;

```

Tipe data bentukan ini digunakan untuk menyimpan data terkait seorang user antara lain; int id, char email, dan char password. int id merupakan angka unik sebagai tanda pengenalan user. Sedangkan email dan password merupakan alamat email serta password untuk alamat email tersebut.

2. ListUser

```

#define CAPACITY_USER 20

typedef struct {
    int number;
    userType data[CAPACITY_USER];
} ListUser;

```

Tipe data bentukan ini digunakan sebagai memori tempat menyimpan (container) elemen dengan tipe data userType sebanyak CAPACITY_USER dari ListUser, selain itu juga terdapat int number untuk menyimpan banyaknya seluruh user yang terdaftar pada program.

Fungsi/Prosedur

1. void CreateListUser(ListUser *l);
Prosedur ini digunakan untuk inisialisasi semua elemen ListUser l dengan MARK. F.S. terbentuk ListUser l kosong dengan kapasitas CAPACITY_USER.
2. void addUser(ListUser *l, userType user);
Prosedur ini digunakan untuk menambahkan user sebagai elemen terakhir ListUser l. F.S. user adalah elemen terakhir l yang baru.
3. int idUser(ListUser l, Word word);
Fungsi ini digunakan untuk mencari id dari berdasarkan email dalam ListUser l. Fungsi ini mengembalikan idUser yang sesuai.

3.12 ADT Pengguna

ADT Pengguna memiliki peran sebagai representasi sebuah struktur data yang menyimpan informasi terkait dengan user. ADT ini digunakan pada implementasi fitur Pengguna sebab pada fitur tersebut diperlukan manajemen user pada program dimana terdapat perintah REGISTER, LOGIN, dan LOGOUT. Terdapat 1 tipe data bentukan dan 5 fungsi pada ADT ini

yang masing-masing dijelaskan spesifikasinya pada file header *pengguna.h*. Tipe data bentukan dan beberapa fungsi utama pada ADT ini antara lain,

Tipe Data Bentukan

1. `activeUser`

```
typedef struct {  
    int id;  
    char* email;  
} activeUser;
```

Tipe data bentukan ini digunakan sebagai tempat menyimpan data terkait seorang user, nantinya tipe data ini akan membantu dalam proses manajemen user aktif pada program. Terdapat `int id` untuk menyimpan id user dan `char email` menyimpan alamat email yang terkait dengan id user tersebut.

Fungsi/Prosedur

1. `void registerUser();`
Prosedur ini digunakan untuk proses registrasi user dengan alamat email baru pada program.
2. `void loginUser();`
Prosedur ini digunakan untuk proses login user dengan alamat email yang sudah terdaftar pada program.
3. `void LogoutUser();`
Prosedur ini digunakan untuk proses logout user dengan alamat email yang saat ini sedang login pada program.

4 Program Utama

Program utama terletak pada file `main.c` yang dapat dikompilasi dengan perintah `make build` dan dapat dijalankan dengan perintah `make run` pada CLI Terminal. Program kemudian meminta masukan nama folder konfigurasi untuk di load ke dalam program. Setelah load berhasil, pengguna dapat melakukan registrasi, login, atau keluar. Setelah pengguna melakukan login, program akan menampilkan menu utama yakni;

- `BUAT_DRAFT` : Fitur ini digunakan untuk membuat draft email
- `INBOX` : Fitur ini digunakan untuk melihat inbox user yang sedang login
- `BALAS_EMAIL` : Fitur ini digunakan untuk membalas email yang dikirimkan dari user lain
- `STATUS_KEPENTINGAN` : Fitur ini digunakan untuk melihat kepentingan tiap user dengan

- NOTIFIKASI : sesama user lainnya
: Fitur ini digunakan untuk melihat notifikasi email yang diterima dan belum dibaca
- SAVE : Fitur ini digunakan untuk menyimpan data terkini dari program
- LOGOUT : Fitur ini digunakan untuk keluar dari akun yang sedang login

Pengguna cukup memasukkan <FITUR>; pada CLI untuk memilih fitur apa yang ingin digunakan lalu program akan mengarahkan pengguna ke fitur tersebut dan selanjutnya pengguna bisa langsung menggunakan fitur yang dipilih.

5 Algoritma-Algoritma Menarik

5.1 N Ary Tree dalam Representasi Binary Tree

```
// Fungsi untuk membentuk emailtree dengan menambah semua reply ke subjek utama
void addReplies(Tree node, ListEmail listEmail) {
    for (int i = 0; i < listEmail.number; i++) { // Mengecek setiap email pada listEmail
        if (listEmail.data[i].reply == node->info) { // Jika email merupakan reply dari email yang sedang dicek
            if (listEmail.data[i].idPenerima == user.id || listEmail.data[i].idCC == user.id) {
                Address newNode = newTreeNode(listEmail.data[i].id);

                if (newNode != NULL) {
                    if (LEFT(node) == NULL) { // Jika email belum memiliki balasan sama sekali
                        LEFT(node) = newNode;
                        addReplies(newNode, listEmail);
                    } else { // Jika email sudah memiliki balasan
                        Address nextNode = LEFT(node);

                        while (RIGHT(nextNode) != NULL) {
                            nextNode = RIGHT(nextNode);
                        }

                        RIGHT(nextNode) = newNode;
                        addReplies(newNode, listEmail);
                    }
                }
            }
        }
    }
}
```

Algoritma `addReplies` melakukan perulangan pada array `listEmail` yang elemennya berupa `emailType` yang kemudian membangun *n* ary tree dalam representasi binary tree jika elemen email pada `listEmail` merupakan balasan dari email mula-mula. Misal node 'c' adalah *child* paling kiri dari sebuah node dalam *n* ary tree sehingga dalam representasi binary tree, node 'c' merupakan *child* kiri dari node yang sama. Sibling sebelah kanan dari 'c' pada *n* ary tree adalah *child* kanan dari node 'c' pada binary tree.

Algoritma unik ini digunakan pada implementasi menu baca pesan dan balas pesan. Pada menu baca pesan, *n* ary tree digunakan membentuk struktur email balasan yang hierarkis sehingga dapat menampilkan email-email dari email mula-mula sampai email balasan terakhir. Selain itu, menu baca pesan membutuhkan *n* ary tree untuk menentukan nomor reply email

balasan. Nomor reply balasan dihitung dari balasan pertama terhadap email mula-mula sampai email balasan terakhir terkait subjek email mula-mula.

Struktur data binary tree dan n ary tree merupakan struktur data hierarkis yang memanfaatkan atribut child pada tree. Namun, keduanya memiliki perbedaan utama dalam jumlah anak dari node. Dengan mengonversi n-ary tree menjadi binary tree, kita bisa menyederhanakan struktur pohon yang kompleks menjadi lebih sederhana. Ini mempermudah implementasi algoritma seperti traversal yang efisien dan algoritma pencarian cepat (Binary Search Tree) yang sudah dioptimalkan untuk binary tree.

6 Data Test

6.1 Insialisasi

Data test ini digunakan untuk pengujian fitur Inisialisasi pada program utama. Hasil akhir yang diharapkan dari fitur ini adalah semua file config terbaca dan disimpan dalam program. Apabila pengguna memasukkan input salah yakni memasukkan nama folder yang tidak benar, maka program akan diterminasi.

```
Masukkan folder konfigurasi untuk dimuat: config;
Memuat konfigurasi Purry Mail...
Memuat konfigurasi Purry Mail...
Memuat konfigurasi Purry Mail...
Silakan REGISTER/LOGIN untuk memulai!
Masukkan Opsi Masuk Program:
--- REGISTER
--- LOGIN
--- QUIT
```

Gambar 1.1 User Memasukkan Input Benar

```
Masukkan folder konfigurasi untuk dimuat: folderpalsu;
Gagal memuat file config dari folderpalsu.
Exiting...
```

Gambar 1.2 User Memasukkan Input Tidak Benar

6.2 Pengguna

Data test ini digunakan untuk pengujian fitur Pengguna pada program utama. Apabila user memasukkan REGISTER maka program akan meminta email dan password. Jika email tidak memenuhi ketentuan regex, panjang, atau email sudah terdaftar maka program akan meminta masukkan lagi. Jika password melebihi panjang maksimum maka program akan meminta masukkan lagi. Email dan Password berhasil didaftarkan pada program apabila semua ketentuan terpenuhi.

```
Masukkan email address: test<>@0*.1%;
Email tidak valid!
```

Gambar 2.1 Email memiliki regex yang tidak valid

```
Masukkan email address: email3;  
Email tidak valid!
```

Gambar 2.2 Email sudah terdaftar

```
Masukkan email address: panjang255aaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa@ow.ca;  
Panjang email maksimal 255 karakter.
```

Gambar 2.3 Email melebihi panjang maksimal

```
Masukkan email address: test@ow.ca;
Masukkan kata sandi: sandi20aaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaa;
Gagal membuat password :(
    Panjang password maksimal adalah 20 Karakter.
```

Gambar 2.4 Password melebihi panjang maksimal

```
Masukkan email address: testing@ow.ca;
Masukkan kata sandi: testing;
Akun berhasil didaftarkan! Silakan LOGIN untuk memulai
menggunakan program.
Masukkan Opsi Masuk Program:
    --- REGISTER
    --- LOGIN
    --- QUIT
```

Gambar 2.5 Masukan Email dan Password Valid

Apabila user memasukkan LOGIN maka program akan meminta email dan password. Jika email dan password yang dimasukan tidak benar, maka program akan meminta masukan kembali. Masukan Email dan Password yang benar akan membuat program menampilkan menu utama.

```
Masukkan email address: testingsalah@ow.ca;
Email belum terdaftar!
```

Gambar 2.6 Masukan Email Belum Terdaftar

```
Masukkan email address: testing@ow.ca;
Masukkan password: testingsalah;
Password salah!
```

Gambar 2.7 Masukan Password Salah

```
Masukkan email address: testing@ow.ca;
Masukkan password: testing;
Login berhasil, selamat datang di PurryMail!
```

Gambar 2.8 Login Berhasil

6.3 Draft Email

Data test ini digunakan untuk pengujian fitur Draft Email pada program utama. Apabila pengguna memasukkan BUAT_DRAFT maka program akan masuk ke menu pembuatan draft jika pengguna belum memiliki draft. Apabila pengguna sudah memiliki draft maka program akan mengirimkan pesan tanda sudah ada draft.

```
>> BUAT_DRAFT;
Maaf, sudah ada draft terbuat! Silahkan finalisasi draft yang sudah dibuat.
```

Gambar 3.1 Perintah BUAT_DRAFT Namun Sudah Ada Draft

Jika pengguna memasukkan alamat email penerima yang tidak valid/kosong maka program akan meminta masukan kembali. Hal yang sama juga akan terjadi pada saat memasukkan alamat email CC namun pada email CC dimungkinkan untuk kosong.

```
=====
SELAMAT DATANG DI MENU DRAFT EMAIL
Masukkan Email Penerima : EMAILINVALID@ow.ca;
Pengguna email dengan alamat EMAILINVALID@ow.ca tidak ditemukan.
Masukkan "STOP" untuk force stop Pembuatan Draft Email.
Masukkan Email Penerima : ;
Tidak ada email yang dituju, silakan ulangi.
Masukkan "STOP" untuk force stop Pembuatan Draft Email.
```

Gambar 3.2 Email Penerima Tidak Valid/Kosong

```
Masukkan Email Penerima : aaa@b.cc;
Masukkan Penerima CC : █
```

Gambar 3.3 Email Penerima Valid

```
Masukkan Penerima CC : EMAILINVALID@ow.ca;
Pengguna email dengan alamat EMAILINVALID@ow.ca tidak ditemukan.
Masukkan "STOP" untuk force stop Pembuatan Draft Email.
Masukkan Penerima CC : testing2@ow.ca;
Penerima CC tidak boleh sama dengan penerima pesan.
Masukkan "STOP" untuk force stop Pembuatan Draft Email.
```

Gambar 3.4 Email CC Tidak Valid

```
Masukkan Penerima CC : ;
Masukkan Subjek : █
```

Gambar 3.5 Email CC Kosong

Selanjutnya program akan meminta masukan subjek, jika subjek kosong/terdapat newline maka program akan meminta masukan kembali. Selanjutnya program akan meminta masukan

body, jika body kosong/melebihi batas maksimal karakter maka program akan meminta masukan kembali. Apabila semua input valid maka draft berhasil disimpan oleh program.

```
Masukkan Subjek : ;
Subjek tidak valid karena kosong.
Masukkan "STOP" untuk force stop Ubah Draft Email.
Masukkan Subjek : test
newline;
Subjek tidak valid karena melebihi batas 250 karakter / melebihi batas 1 baris.
Masukkan "STOP" untuk force stop Ubah Draft Email.
```

Gambar 3.6 Subjek Tidak Valid

```
Masukkan Body : ;
Body tidak valid karena kosong.
Masukkan "STOP" untuk force stop Ubah Draft Email.
```

Gambar 3.7 Body Kosong

[illegible]

Gambar 3.8 Body Melebihi Batas Maksimum Karakter

```

=====
                        DRAFT EMAIL
=====
SELAMAT DATANG DI MENU DRAFT EMAIL
Masukkan Email Penerima : aaa@b.cc;
Masukkan Penerima CC : ;
Masukkan Subjek : testing;
Masukkan Body : testing;
Draft Email berhasil dibuat.

```

Gambar 3.9 Draft Berhasil Dibuat

Jika pengguna memasukkan KIRIM_DRAFT maka program akan mengirimkan draft tersebut dan program akan otomatis keluar dari menu pembuatan draft kembali ke menu utama. Jika pengguna memasukkan UBAH_DRAFT maka program akan meminta kembali mengisi data

terkait draft email dan apabila berhasil maka draft terbaru akan disimpan oleh program. Jika pengguna memasukkan LIHAT_DRAFT maka program akan menampilkan draft email terkini.

```
>> KIRIM_DRAFT;  
Draft Email berhasil dikirim!
```

Gambar 3.10 Perintah KIRIM_DRAFT

```
>> UBAH_DRAFT;  
Masukkan Email Penerima : aaa@b.cc;  
Masukkan Penerima CC : ;  
Masukkan Subjek : testing;  
Masukkan Body : revisi testing;  
Draft Email berhasil diubah.
```

Gambar 3.11 Perintah UBAH_DRAFT

```
>> LIHAT_DRAFT;  
Email Penerima : aaa@b.cc  
Penerima CC : <Tidak Ada>  
Subjek : testing  
Body : revisi testing
```

Gambar 3.12 Perintah LIHAT_DRAFT

Jika pengguna memasukkan UNDO maka program akan mengembalikan versi sebelumnya dari draft email, apabila tidak ada versi sebelumnya maka program akan memberikan pesan tanda tidak bisa undo. Jika pengguna memasukkan REDO maka program akan mengembalikan versi sebelumnya dari UNDO pada draft email, apabila tidak ada maka program akan memberikan pesan tanda tidak bisa REDO. Jika pengguna memasukkan BATAL maka program akan menghapus draft email dan kembali ke menu utama. Apabila pengguna memasukkan perintah yang tidak valid dalam menu pembuatan draft maka program akan memberikan pesan tanda dan meminta masukan kembali.

```
>> UNDO;  
Perubahan berhasil di-undo.
```

Gambar 3.14 Perintah UNDO Berhasil

```
>> UNDO;  
Belum ada perubahan yang dapat di-undo.
```

Gambar 3.15 Tidak Ada Versi Sebelumnya

```
>> REDO;  
Perubahan berhasil di-redo.
```

Gambar 3.16 Perintah REDO Berhasil

```
>> REDO;  
Belum ada perubahan yang dapat di-redo.
```

Gambar 3.17 Tidak Ada Perubahan Versi

```
>> BATAL;  
Draft Email berhasil dibatalkan!
```

Gambar 3.18 Perintah BATAL


```
>> PERINTAHINVALID;
Perintah Tidak Valid. Masukkan "BATALL" untuk kembali ke menu utama.
```

Gambar 3.19 Perintah Tidak Valid Pada Menu Pembuatan Draft

6.4 Inbox Utama Email

Data test ini digunakan untuk pengujian fitur Inbox pada program utama. Apabila user memasukkan INBOX maka program akan masuk ke menu inbox. Jika pengguna memasukkan DAFTAR_INBOX pada menu inbox maka program akan memunculkan daftar email berurut dari yang terbaru dengan paging sesuai config.

```
>> INBOX;
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                INBOX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Selamat datang di Menu Inbox!
Masukkan Menu:
    --- DAFTAR_INBOX
    --- LANJUT
    --- SEBELUM
    --- BACA_PESAN EMAILXXX
    --- BALAS_PESAN EMAILXXX
    --- DAFTAR_STARRED
    --- LANJUT_STAR
    --- SEBELUM_STAR
    --- STAR EMAILXXX
    --- UNSTAR EMAILXXX
    --- KELUAR

XXX merupakan Email ID yang ingin diakses yaa
Jangan lupa sertakan ';' di akhir yaa ~~~
```

Gambar 4.1 Program Masuk ke Menu Inbox

```
>> DAFTAR_INBOX;
[----- Anda memiliki 6 pesan pada Inbox -----]
[-----[ Daftar Inbox ]-----]
| Email ID | Subject          | Pengirim      | Status | Timestamp          | Starred |
|-----|-----|-----|-----|-----|-----|
| EMAIL016 | test4            | aaa@b.cc      | Unread | 2024-12-13 @ 18.18.09 | No      |
| EMAIL015 | test3            | aaa@b.cc      | Unread | 2024-12-13 @ 18.17.11 | No      |
| EMAIL014 | test2            | aaa@b.cc      | Unread | 2024-12-13 @ 18.16.42 | No      |
| EMAIL013 | test1            | aaa@b.cc      | Unread | 2024-12-13 @ 18.16.00 | No      |
| EMAIL012 | test             | testing1@ow.ca | Unread | 2024-12-13 @ 18.11.43 | No      |
|-----|-----|-----|-----|-----|-----|
[----- Page [1]/[2] -----]
[-----]
```

Gambar 4.2 Perintah DAFTAR_INBOX

Jika pengguna memasukkan LANJUT pada menu inbox maka program akan menampilkan daftar email selanjutnya, apabila tidak ada email lagi maka program akan menampilkan pesan tanda halaman terakhir. Jika pengguna memasukkan SEBELUM pada menu inbox maka program akan menampilkan daftar email sebelumnya, apabila tidak ada email lagi maka program akan menampilkan pesan tanda halaman pertama.

Masukkan perintah dalam mode INBOX: >> SEBELUM;
Sudah merupakan halaman yang paling pertama.

Gambar 4.3 Perintah SEBELUM Pada Halaman Pertama

Masukkan perintah dalam mode INBOX: >> LANJUT;
Sudah merupakan halaman yang paling terakhir.

Gambar 4.4 Perintah LANJUT Pada Halaman Terakhir

```
>> SEBELUM;
Halaman telah berganti ke 1.
[----- Anda memiliki 6 pesan pada Inbox -----]
[-----[ Daftar Inbox ]-----]
| Email ID | Subject | Pengirim | Status | Timestamp | Starred |
|-----|-----|-----|-----|-----|-----|
| EMAIL016 | test4 | aaa@b.cc | Unread | 2024-12-13 @ 18.18.09 | No |
| EMAIL015 | test3 | aaa@b.cc | Unread | 2024-12-13 @ 18.17.11 | No |
| EMAIL014 | test2 | aaa@b.cc | Unread | 2024-12-13 @ 18.16.42 | No |
| EMAIL013 | test1 | aaa@b.cc | Unread | 2024-12-13 @ 18.16.00 | No |
| EMAIL012 | test | testing1@ow.ca | Unread | 2024-12-13 @ 18.11.43 | No |
|-----|-----|-----|-----|-----|-----|
|----- Page [1]/[2] -----|
|-----|-----|-----|-----|-----|-----|
```

Gambar 4.5 Perintah SEBELUM

```
>> LANJUT;
Halaman telah berganti ke 2.
[----- Anda memiliki 6 pesan pada Inbox -----]
[-----[ Daftar Inbox ]-----]
| Email ID | Subject | Pengirim | Status | Timestamp | Starred |
|-----|-----|-----|-----|-----|-----|
| EMAIL011 | testing | testing1@ow.ca | Unread | 2024-12-13 @ 18.11.08 | No |
|-----|-----|-----|-----|-----|-----|
|----- Page [2]/[2] -----|
|-----|-----|-----|-----|-----|-----|
```

Gambar 4.6 Perintah LANJUT

Jika pengguna memasukkan BACA_PESAN <Email ID> pada menu inbox maka program akan menampilkan isi dari email tersebut, apabila Email ID tidak valid maka program akan meminta masukkan kembali. Jika pengguna memasukkan KELUAR pada menu inbox maka program akan keluar dari menu inbox. Jika pengguna memasukkan perintah yang tidak valid pada menu inbox maka program akan meminta masukkan kembali.

```
>> BACA_PESAN EMAIL016;
bottom: 11
top: 16
[-----[ Baca Pesan ]-----]
Inbox ID: EMAIL016
Subject: test4
Pengirim: aaa@b.cc
Timestamp: 2024-12-13 18.18.09
|-----|
test4
|-----|
```

Gambar 4.7 Perintah BACA_PESAN

```
>> BACA_PESAN EMAIL999;
bottom: 11
top: 16
Email tidak ditemukan. Pastikan email yang ingin dibaca
berada pada halaman DAFTAR_INBOX yang sedang dibuka.
```

Gambar 4.8 Perintah BACA_PESAN Dengan Email ID Tidak Valid

```
Masukkan perintah dalam mode INBOX: >> BACA_PESAN AZAZAZAZAZAZ;
Perintah tidak valid. Format harus 'EMAILxxx'.
```

Gambar 4.9 Perintah BACA_PESAN Dengan Input Tidak Valid

```
Masukkan perintah dalam mode INBOX: >> PERINTAH_INVALID;
Perintah tidak valid dalam mode INBOX.
```

Gambar 4.10 Perintah Tidak Valid Pada Menu Inbox

```
Masukkan perintah dalam mode INBOX: >> KELUAR;
Keluar dari mode INBOX.
```

Gambar 4.11 Perintah KELUAR Pada Menu Inbox

6.5 Balas Email

Data test ini digunakan untuk pengujian fitur BALAS_EMAIL pada program utama. Pengguna harus masuk kedalam menu INBOX terlebih dahulu untuk bisa menggunakan fitur BALAS_PESAN. Jika pengguna memasukkan BALAS_PESAN <Email ID> maka program akan masuk ke menu BUAT_DRAFT dengan penerima adalah pengirim dan subyek adalah Re[nomor_reply] Subyek dari email yang di balas. Apabila Email ID tidak valid maka program akan meminta masukkan kembali.

```
>> INBOX;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                               INBOX
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Selamat datang di Menu Inbox!
Masukkan Menu:
  --- DAFTAR_INBOX
  --- LANJUT
  --- SEBELUM
  --- BACA_PESAN EMAILXXX
  --- BALAS_PESAN EMAILXXX
  --- DAFTAR_STARRED
  --- LANJUT_STAR
  --- SEBELUM_STAR
  --- STAR EMAILXXX
  --- UNSTAR EMAILXXX
  --- KELUAR
XXX merupakan Email ID yang ingin diakses yaa
Jangan lupa sertakan ';' di akhir yaa ~~~~
```

Gambar 5.1 Menu INBOX

```
>> BALAS_PESAN EMAIL999;
Email tidak ditemukan. Pastikan email yang ingin dibaca
berada pada halaman DAFTAR_INBOX yang sedang dibuka.
```

Gambar 5.2.a Perintah BALAS_PESAN Dengan Email ID Tidak Valid

```
>> BALAS_PESAN INVALID;
Perintah tidak valid dalam mode INBOX.
```

Gambar 5.2.b Perintah BALAS_PESAN Dengan Argumen Tidak Valid

```
>> BALAS_PESAN EMAIL006;

=====
SELAMAT DATANG DI MENU DRAFT BALAS EMAIL
Masukkan Email Penerima : 13523021_Raihan@std.stei.itb.ac.id
Masukkan Penerima CC : ;
Masukkan Subjek : Re[1]: Subyek2
Masukkan Body : test;
Draft Balas Email berhasil dibuat.
```

Gambar 5.3 Perintah BALAS_PESAN Dengan Email ID Valid (Menuju Fitur BUAT_DRAFT)

Jika pengguna memasukkan BACA_PESAN <Email ID> maka program akan menampilkan isi dari email tersebut dan email-email balasan sebelumnya, apabila Email ID tidak valid maka program akan meminta masukkan kembali.

```
>> BACA_PESAN EMAIL999;
Email tidak ditemukan. Pastikan email yang ingin dibaca
berada pada halaman DAFTAR_INBOX yang sedang dibuka.
```

Gambar 5.4 Perintah BACA_PESAN Dengan Email ID Tidak Valid

```
>> BACA_PESAN EMAIL009;
[-----[ Baca Pesan ]-----]
Inbox ID: EMAIL009
Subject: coba trial
Pengirim: 13523021_Raihan@std.stei.itb.ac.id
Timestamp: 2024-12-11 09.30.43
[-----]

[coba trial]
[-----]
halo ini nama saya
coba coba coba
[-----]
```

Gambar 5.5 Perintah BACA_PESAN Dengan Email ID Valid

6.6 Status Kepentingan

Data test ini digunakan untuk pengujian fitur Status Kepentingan pada program utama. Apabila user memasukkan STATUS_KEPENTINGAN maka program masuk ke dalam menu status kepentingan dan akan menampilkan status kepentingan tiap user terhadap user lainnya dalam bentuk tabel sesuai format. Untuk keluar dari menu status kepentingan, user harus memasukkan perintah KELUAR.

ID	01	02	03	04	05	06	07	08	09	10	11
01	00	03	07	00	00	00	00	00	06	00	00
02	00	00	00	00	00	00	00	00	00	00	00
03	00	00	00	00	00	00	00	00	00	00	00
04	00	00	00	00	00	00	00	00	00	00	00
05	00	00	00	00	00	00	00	00	00	00	00
06	00	00	00	00	00	00	00	00	00	00	00
07	00	00	00	00	00	00	00	00	00	00	00
08	00	00	00	00	00	00	00	00	00	00	00
09	00	00	00	00	00	00	00	00	01	00	04
10	00	00	00	00	00	00	00	00	00	02	02
11	00	00	00	00	00	00	00	00	00	00	00

Masukkan perintah KELUAR untuk kembali ke Menu Utama

Gambar 6.1 Perintah STATUS_KEPENTINGAN

```
>> KELUAR;
Keluar dari Status Kepentingan...
```

Gambar 6.2 Perintah KELUAR

6.7 Notifikasi

Data test ini digunakan untuk pengujian fitur Notifikasi pada program utama. Apabila user memasukkan NOTIFIKASI maka program akan menampilkan daftar email dengan status belum dibaca yang ditujukan untuk user tersebut terurut berdasarkan Kepentingan dan Waktu secara menaik. Apabila tidak ada notifikasi, maka program akan menampilkan pesan tanda tidak ada notifikasi. Fitur Notifikasi ini juga secara otomatis dijalankan setiap kali seorang user Login.

```
>> NOTIFIKASI;
Notifikasi:
2024-10-17 19.00.37 [IMPORTANT] H4k3r Vacancy at OWCA
2024-10-18 01.00.37 [IMPORTANT] Subyek2
2024-10-18 01.30.37 [IMPORTANT] Subyek2
2024-10-18 02.16.37 [IMPORTANT] Subyek2
2024-10-18 03.00.37 [IMPORTANT] Subyek2
2024-10-18 05.00.37 [IMPORTANT] Subyek2
2024-10-18 05.10.37 [IMPORTANT] Subyek2
2024-10-18 09.00.37 [IMPORTANT] Subyek2
2024-10-18 12.00.37 [IMPORTANT] Subyek2
2024-10-18 16.00.37 [IMPORTANT] Subyek2
2024-10-18 17.00.37 [IMPORTANT] Subyek2
2024-12-11 11.30.43 coba trial
```

Gambar 7.1 Perintah NOTIFIKASI

```
>> NOTIFIKASI;
Tidak ada notifikasi.
```

Gambar 7.2 Tidak Ada Notifikasi


```

Masukkan folder konfigurasi untuk dimuat: config;
Memuat konfigurasi Purry Mail...
Memuat konfigurasi Purry Mail...
Memuat konfigurasi Purry Mail...
Silakan REGISTER/LOGIN untuk memulai!
Masukkan Opsi Masuk Program:
    --- REGISTER
    --- LOGIN
    --- QUIT

```

Gambar 8.2 Perintah LOAD Awal Menjalankan Program

```

Masukkan folder konfigurasi untuk dimuat: folderpalsu;
Gagal memuat file config dari folderpalsu.
Exiting...

```

Gambar 8.3 Perintah LOAD Masukan Folder Yang Salah

Jika pengguna memasukkan perintah SAVE maka program akan meminta nama folder untuk tempat menyimpan data saat ini, apabila nama folder belum ada maka program akan membuat folder baru dengan nama tersebut dan apabila nama folder sudah ada maka program akan menimpa data yang ada pada folder tersebut.

```

>> SAVE;
Masukkan nama folder penyimpanan: config;
File konfigurasi telah disimpan!

```

Gambar 8.4 Perintah SAVE Dengan Folder Lama

```

>> SAVE;
Masukkan nama folder penyimpanan: newfolder;
Folder berhasil dibuat!
File konfigurasi telah disimpan!

```

Gambar 8.5 Perintah SAVE Dengan Folder Baru

6.9 Starred Email

Data test ini digunakan untuk pengujian fitur Starred Email pada program utama. Untuk menggunakan fitur ini pengguna harus masuk ke dalam menu inbox dengan memasukkan perintah INBOX. Apabila pengguna memasukkan perintah STAR <Email ID> maka program akan menandai email tersebut dengan starred, jika argumen/email tidak valid maka program akan meminta masukan kembali dari pengguna. Apabila email sudah di starred maka program akan memberikan pesan tanda email sudah di starred. Apabila pengguna memasukkan perintah UNSTAR <Email ID> maka program akan menghilangkan tanda starred dari email tersebut. Apabila argumen/email tidak valid atau email sudah tidak di starred maka program akan memberikan respon yang sama seperti diatas. Apabila pengguna memasukkan perintah DAFTAR_STARRED maka program akan menampilkan INBOX yang hanya memiliki email starred.

```
>> INBOX;
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                INBOX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Selamat datang di Menu Inbox!
Masukkan Menu:
    --- DAFTAR_INBOX
    --- LANJUT
    --- SEBELUM
    --- BACA_PESAN EMAILXXX
    --- BALAS_PESAN EMAILXXX
    --- DAFTAR_STARRED
    --- LANJUT_STAR
    --- SEBELUM_STAR
    --- STAR EMAILXXX
    --- UNSTAR EMAILXXX
    --- KELUAR
XXX merupakan Email ID yang ingin diakses yaa
Jangan lupa sertakan ';' di akhir yaa ~~~
```

Gambar 9.1 Menu Inbox

```
>> STAR EMAIL999;
Email tidak ditemukan. Pastikan email yang ingin dibaca
berada pada halaman DAFTAR_INBOX yang sedang dibuka.
```

Gambar 9.2 Perintah STAR Argumen/Email Tidak Valid

```
>> STAR EMAIL009;
EMAIL009 sudah berada di STARRED.
```

Gambar 9.3 Perintah STAR Email Yang Sudah di Starred

```
>> STAR EMAIL006;
EMAIL006 berhasil dipindahkan ke STARRED.
```

Gambar 9.4 Perintah STAR Berhasil

```
>> UNSTAR EMAIL999;
Email tidak ditemukan. Pastikan email yang ingin dibaca
berada pada halaman DAFTAR_INBOX yang sedang dibuka.
```

Gambar 9.5 Perintah UNSTAR Argumen/Email Tidak Valid

```
>> UNSTAR EMAIL003;
EMAIL003 tidak berada di daftar STARRED.
```

Gambar 9.6 Perintah UNSTAR Email Yang Sudah Tidak di Starred

```
>> UNSTAR EMAIL006;
EMAIL006 berhasil dihapus dari daftar STARRED.
```

Gambar 9.7 Perintah UNSTAR Berhasil

```
>> DAFTAR_STARRED;
[----- Anda memiliki 4 pesan pada Inbox -----]
[-----[ Daftar Starred ]-----]
[ Email ID | Subject | Pengirim | Status | Timestamp |
[-----]
[ EMAIL009 | coba trial | 13523021_Raihan | Unread | 2024-12-11 @ 09.30.43 |
[ EMAIL004 | Subyek2 | 13523021_Raihan | Read | 2024-10-18 @ 20.00.00 |
[ EMAIL002 | Subyek2 | 13523021_Raihan | Read | 2024-10-18 @ 20.00.00 |
[ EMAIL001 | H4k3r Vacancy at 0 | 13523021_Raihan | Read | 2024-10-17 @ 19.00.00 |
[-----]
[----- Page [1]/[1] -----]
[-----]
```

Gambar 9.8 Perintah DAFTAR_STARRED

7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Inisialisasi	Menguji apakah fitur sudah berjalan dengan benar	Sama seperti pada Data Test	Inisialisasi	Program memberikan hasil sesuai dengan spesifikasi pada Data Test	Sesuai dengan hasil yang diinginkan (Gambar pada data test)
2	Pengguna	Menguji apakah fitur sudah berjalan dengan benar	Sama seperti pada Data Test	Pengguna	Program memberikan hasil sesuai dengan spesifikasi pada Data Test	Sesuai dengan hasil yang diinginkan (Gambar pada data test)
3	Draft Email	Menguji apakah fitur sudah berjalan dengan benar	Sama seperti pada Data Test	Draft Email	Program memberikan hasil sesuai dengan spesifikasi pada Data Test	Sesuai dengan hasil yang diinginkan (Gambar pada data test)
4	Inbox Utama Email	Menguji apakah fitur sudah berjalan dengan benar	Sama seperti pada Data Test	Inbox Utama Email	Program memberikan hasil sesuai dengan spesifikasi pada Data Test	Sesuai dengan hasil yang diinginkan (Gambar pada data test)
5	Balas Email	Menguji apakah fitur sudah berjalan dengan benar	Sama seperti pada Data Test	Balas Email	Program memberikan hasil sesuai dengan spesifikasi pada Data Test	Sesuai dengan hasil yang diinginkan (Gambar pada data test)

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
6	Status Kepentingan	Menguji apakah fitur sudah berjalan dengan benar	Sama seperti pada Data Test	Status Kepentingan	Program memberikan hasil sesuai dengan spesifikasi pada Data Test	Sesuai dengan hasil yang diinginkan (Gambar pada data test)
7	Notifikasi	Menguji apakah fitur sudah berjalan dengan benar	Sama seperti pada Data Test	Notifikasi	Program memberikan hasil sesuai dengan spesifikasi pada Data Test	Sesuai dengan hasil yang diinginkan (Gambar pada data test)
8	Save & Load	Menguji apakah fitur sudah berjalan dengan benar	Sama seperti pada Data Test	Save & Load	Program memberikan hasil sesuai dengan spesifikasi pada Data Test	Sesuai dengan hasil yang diinginkan (Gambar pada data test)
9	Starred Email	Menguji apakah fitur sudah berjalan	Sama seperti pada Data Test	Starred Email	Program memberikan hasil sesuai dengan spesifikasi pada Data Test	Sesuai dengan hasil yang diinginkan (Gambar pada data test)

8 Pembagian Kerja dalam Kelompok

Anggota	Pengerjaan
Shannon Aurellius Anastasya Lie (13523019)	ADT List Statik, ADT List Dinamik, Perintah, Inbox Utama, Starred Email, Laporan
Muhammad Raihan Nazhim Oktana (13523021)	ADT Tree, Draft Email, Laporan
Joel Hotlan Haris Siahaan (13523025)	ADT Stack, ADT Priority Queue (Queue), Balas Email, Laporan
Najwa Kahani Fatima (13523043)	ADT Mesin Karakter & Kata, Inisialisasi, Pengguna, Save & Load, Laporan
Muhammad Iqbal Haidar (13523111)	ADT Datetime, Status Kepentingan, Notifikasi, Laporan

9 Lampiran

9.1 Deskripsi Tugas Besar 1

"WIU WIU 🚨," suara sirene bergema di sekujur kantor OWCA. Layar-layar monitor berkedip-kedip, menampilkan tulisan "SISTEM DIRETAS". Agen-agen yang biasanya tenang kini berlarian ke sana kemari, berusaha mencari cara untuk mengamankan informasi rahasia milik OWCA. Suasana kantor saat itu benar-benar kacau.

Semua masalah ini berawal dari aplikasi klien surel yang digunakan OWCA, ZMail, sebuah perangkat lunak yang populer digunakan untuk komunikasi internal. OWCA mengandalkan ZMail untuk menyampaikan informasi penting kepada agen-agen mereka. Namun, tak disangka, sebuah kerentanan kritis ditemukan dalam ZMail—celah keamanan yang memungkinkan pihak luar mengakses pesan rahasia OWCA. Tuan Asep Spakbor pun memanfaatkan celah ini untuk mengetahui informasi rahasia OWCA. Dengan informasi tersebut, ia pun berhasil meretas sistem OWCA.

Menanggapi masalah ini, Mr. Monogram pun memerintahkan agen untuk berhenti menggunakan ZMail. Mr. Monogram pun memerintahkan Purry untuk membuat aplikasi klien surel pengganti—PurryMail.



Mr. Monogram berbicara dengan Purry

Namun, Purry tidak bisa memprogram kode. Oleh karena itu, bantulah Purry untuk membuat aplikasi PurryMail!

9.2 Notulen Rapat

Notulen Asistensi 1:

- Boleh import time buat ambil local date time aja (local date time komputer).
- ADT time menyimpan data tahun, bulan, hari, jam, menit, dan detik.
- ADT time digunakan pada notifikasi dan untuk timestamp.
- ADT queue buat priority (contohnya pada bagian notifikasi).
- ADT tree kurang lebih serupa dengan binary tree (meskipun perlu ada modifikasi dikit agar bisa dipakai di spek yg ada).
- Kalau bisa timeline kasarnya saat asistensi 2 dilaksanakan nanti sudah selesai alur program utama.
- Cara baca file windows itu CRLF, kalau mac & linux (UNIX) itu LF.
- Bahasa C cukup dekat dengan bahasa mesin, sehingga banyak yang manual. Oleh karena itu, biasanya jika beda OS maka akan beda perlakuan juga.
- Kedua hal diatas bisa menyebabkan muncul bug aneh ketika integrate beda OS, oleh karena itu, gunakan WSL untuk membiasakan juga.
- Nama kelompok menyesuaikan nama repository di Github saja, tidak perlu membuat nama kelompok yang baru / unik.

Notulen Asistensi 2:

- Pada pembuatan draft email, subyek dibatasi maksimal 1 baris dengan banyak karakter maksimal adalah 250 karakter dan tidak kosong, sedangkan untuk body banyak baris dibebaskan tapi tetap dibatasi oleh karakter maksimal sebanyak 1000 karakter dan tidak kosong.
- Saat melakukan ubah draft, maka seluruh kemungkinan untuk redo akan dihapus dan hasil ubah menjadi versi terbaru, sementara kemungkinan undo lainnya akan tetap tersedia.
- Saat load diberi peringatan untuk save config sebelumnya saja sebagai handling case config yang sebelumnya belum tersimpan dengan baik.
- Pada proses balas email, tree yang dibuat perlu representasi n-ary dalam binary jika email yang sama di reply berkali-kali, terutama jika di-reply lebih dari 2 kali. Penomoran mengikuti banyak node pada treenya.
- Pastikan seluruh program sudah di integrasi dan dicoba dijalankan di LINUX / UNIX, mungkin dengan WSL, karena terkadang perlu beberapa penyesuaian yang mungkin memakan waktu.

9.3 Log Activity Anggota Kelompok

Tanggal	12/11/2024	18/11/2024	25/11/2024	02/12/2024	09/11/2024
	-	-	-	-	-
	17/11/2024	24/11/2024	01/12/2024	08/12/2024	13/11/2024

Shannon Aurellius Anastasya Lie (13523019)	ADT List Statik, ADT List Dinamik,	Inbox	Inbox	Inbox	Laporan
Muhammad Raihan Nazhim Oktana (13523021)	ADT Tree	Draft Email	Draft Email	Debugging	Laporan & Debugging
Joel Hotlan Haris Siahaan (13523025)	ADT Priority Queue, ADT Stack		Balas Email	Balas Email	Laporan
Najwa Kahani Fatima (13523043)	ADT Mesin Karakter & Kata,	Inisialisasi	Pengguna	Save & Load	Laporan, Debugging, Integrasi Program
Muhammad Iqbal Haidar (13523111)	ADT Datetime		Status Kepentingan	Notifikasi	Laporan