Modul 5

Code

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<malloc.h>
#define MAX 1000

void header(){
    printf("\t\t\t\tMade by\n\t\t\t\tPristian Budi Dharmawan - 2501983105\n");
    printf("\t\t\t\t\t\tVer. 2.03.10");
}

int front = -1, rear = -1;
char flight_airlines[MAX][100], plane_airlines[100], flight_number[MAX][10], plane_number[10];

struct linkedList{
    char fl_airlines[100], fl_number[10];
    struct linkedList *next;
};

struct linkedList *front1, *rear1;

void front_rearArray(){
    if(front == -1 || front > rear){
        printf("EMPTY!!!\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } else{
        //Print the front and the rear
        printf("The Front of The Queue\n");
        printf("AIRLINE\t\t: %s\n", flight_airlines[front]);
        printf("FLIGHT NUMBER\t: %s\n\n", flight_number[front]);
        printf("The Rear of The Queue\n");
        printf("AIRLINE\t\t: %s\n", flight_airlines[rear]);
        printf("FLIGHT NUMBER\t: %s\n\n", flight_number[rear]);
        printf("Press ENTER to continue..."); getchar(); getchar();
    }
}

void enqueueArray(){
    //Input array to the queue
    printf("Enter your flight number: ");
    scanf("%s", &plane_number); fflush(stdin);
    scanf("%[^\n]", &plane_airlines, printf("Enter your flight airlines: ")); fflush(stdin);

    if(rear == MAX-1){
        printf("\n\nOVERFLOW!!!\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } else if(front == -1 && rear == -1)
        front = rear = 0;
    else{
```

```
        rear++;
    }

    //Filling from behind
    strcpy(flight_number[rear], plane_number);
    strcpy(flight_airlines[rear], plane_airlines);
    printf("\n\t\t===Input data Succeeded!===\n\n");
    printf("Press ENTER to continue..."); getchar();
}

void dequeArray(){
    if(front == -1 || front > rear){
        printf("UNDERFLOW!!!\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } else{
        //Deleting front of the queues
        strcpy(plane_number, flight_number[front]);
        strcpy(plane_airlines, flight_airlines[front]);
        front++;

        //Default front & rear after insertion if there's no more data inside
        if(front > rear)
            front = rear = -1;

        //Additional information
        printf("\t\t===POPPED DATA FROM QUEUES===\n\n");
        printf("AIRLINE\t\t: %s\n", plane_airlines);
        printf("FLIGHT NUMBER\t: %s\n\n", plane_number);
        printf("\t\t===Delete data Succeeded!===\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    }
}

void disArray(){
    int num = 0;
    if(front == -1 || front > rear){
        printf("EMPTY!!!\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } else{
        printf("\t\t=====LIKED SONGS=====\n\n");
        printf("/-------------------------------------------------------\\\n");
        printf("| NO.\t| AIRLINES\t\t| FLIGHT NUMBER\t\t|\n");
        printf("+-------------------------------------------------------+\n");

        //Displaying the list
        for(int i=front; i <= rear; i++){
            printf("| %02d\t| %-21s | %-22s|\n", num+1, flight_airlines[i], flight_number[i]);
            num++;
        }

        printf("\\-------------------------------------------------------/\n");
        printf("\n\t\t=====This is EOF=====\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    }
```

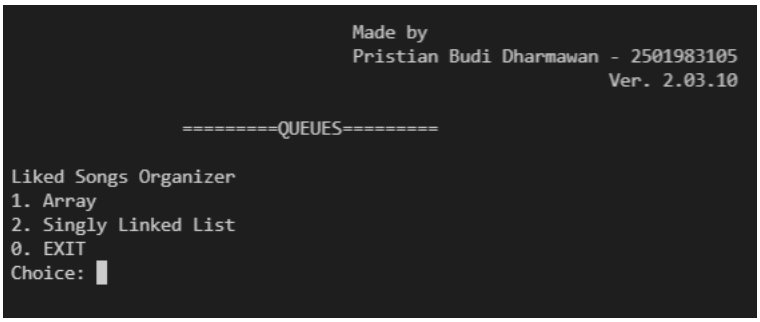
```
}

void front_rearNode(){
    if(front1 == NULL){
        printf("EMPTY!!!\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } else{
        //Print the front and the rear
        printf("The Front of The Queue\n");
        printf("AIRLINE\t\t: %s\n", front1->fl_airlines);
        printf("FLIGHT NUMBER\t: %s\n\n", front1->fl_number);
        printf("The Rear of The Queue\n");
        printf("AIRLINE\t\t: %s\n", rear1->fl_airlines);
        printf("FLIGHT NUMBER\t: %s\n\n", rear1->fl_number);
        printf("Press ENTER to continue..."); getchar(); getchar();
    }
}

void enqueueNode(){
    struct linkedList *ptr = (struct linkedList*)malloc(sizeof(struct linkedList));
    if(ptr == NULL){
        printf("\n\nCan't Push the data\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } else{
        //Input node to the stack
        scanf("%s", &plane_number, printf("Enter your flight number: ")); fflush(stdin);
        scanf("%[^\n]", &plane_airlines, printf("Enter your flight airlines: "));
fflush(stdin);

        *ptr->fl_number = *strcpy(ptr->fl_number, plane_number);
        *ptr->fl_airlines = *strcpy(ptr->fl_airlines, plane_airlines);

        //Assigning the value to the rear of the node
        if(front1 == NULL){
            front1 = ptr;
            rear1 = ptr;
            front1->next = NULL;
            rear1->next = NULL;
        } else{
            rear1->next = ptr;
            rear1 = ptr;
            rear1->next = NULL;
        }

        printf("\n\t\t===Input data Succeeded!===\n\n");
        printf("Press ENTER to continue..."); getchar();
    }
}

void dequeNode(){
    struct linkedList *ptr = front1;

    if(front1 == NULL){
        printf("UNDERFLOW!!!\n\n");
```

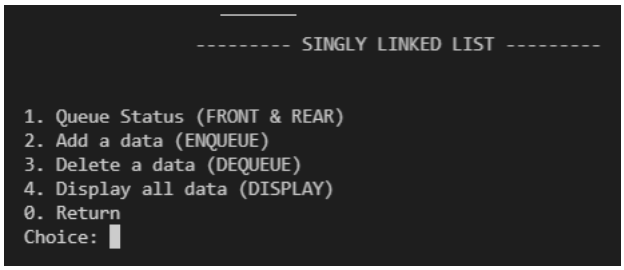
```
        printf("Press ENTER to continue..."); getchar(); getchar();
    } else{
        //Deletion
        front1 = front1->next;
        strcpy(plane_number, ptr->fl_number);
        strcpy(plane_airlines, ptr->fl_airlines);
        //If the front is NULL the rear is also NULL
        if(front1 == NULL) rear1 = NULL;
        free(ptr);

        //Additional Information
        printf("\t\t===DELETED DATA FROM QUEUE===\n\n");
        printf("AIRLINE\t\t: %s\n", plane_airlines);
        printf("FLIGHT NUMBER\t: %s\n\n", plane_number);
        printf("\t\t===Delete data Succeeded!===\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    }
}

void disNode(){
    struct linkedList *ptr = front1;
    int num = 0;

    if(ptr == NULL){
        printf("EMPTY!!!\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } else{
        printf("\t\t=====LIKED SONGS=====\n\n");
        printf("/-------------------------------------------------------\\\n");
        printf("| NO.\t| AIRLINES\t\t| FLIGHT NUMBER\t\t|\n");
        printf("+-------------------------------------------------------+\n");

        //Displaying the list
        while(ptr != NULL){
            printf("| %02d\t| %-21s | %-22s|\n", num+1, ptr->fl_airlines, ptr->fl_number);
            num++;
            ptr = ptr->next;
        }

        printf("\\-------------------------------------------------------/\n");
        printf("\n\t\t=====This is EOF=====\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    }
}

int main(){
    int opt, arrOpt, sllOpt;
    printf("\n");
    do{
        printf("\e[1;1H\e[2J");
        header();
        printf("\n\n\t\t=========QUEUES=========\n\n");
        printf("Liked Songs Organizer\n");
        printf("1. Array\n");
```

```
        printf("2. Singly Linked List\n");
        printf("0. EXIT\n");
        scanf("%d", &opt, printf("Choice: "));
        if(opt == 1){
            do{
                printf("\e[1;1H\e[2J");
                printf("\t\t\t--------- ARRAY ---------\n\n\n");
                printf("1. Queue Status (FRONT & REAR)\n");
                printf("2. Add a data (ENQUEUE)\n");
                printf("3. Delete a data (DEQUE)\n");
                printf("4. Display all data (DISPLAY)\n");
                printf("0. Return\n");
                scanf("%d", &arrOpt, printf("Choice: "));
                switch(arrOpt){
                case 1:
                    printf("\e[1;1H\e[2J"); printf("\n");
                    front_rearArray(); break;
                case 2:
                    printf("\e[1;1H\e[2J"); printf("\n");
                    enqueueArray(); break;
                case 3:
                    printf("\e[1;1H\e[2J"); printf("\n");
                    dequeArray(); break;
                case 4:
                    printf("\e[1;1H\e[2J"); printf("\n");
                    disArray(); break;
                }
            } while(arrOpt != 0);
        } else if(opt == 2){
            do{
                printf("\e[1;1H\e[2J");
                printf("\t\t--------- SINGLY LINKED LIST ---------\n\n\n");
                printf("1. Queue Status (FRONT & REAR)\n");
                printf("2. Add a data (ENQUEUE)\n");
                printf("3. Delete a data (DEQUEUE)\n");
                printf("4. Display all data (DISPLAY)\n");
                printf("0. Return\n");
                scanf("%d", &sllOpt, printf("Choice: "));
                switch(sllOpt){
                case 1:
                    printf("\e[1;1H\e[2J"); printf("\n");
                    front_rearNode(); break;
                case 2:
                    printf("\e[1;1H\e[2J"); printf("\n");
                    enqueueNode(); break;
                case 3:
                    printf("\e[1;1H\e[2J"); printf("\n");
                    dequeNode(); break;
                case 4:
                    printf("\e[1;1H\e[2J"); printf("\n");
                    disNode(); break;
                }
            } while(sllOpt != 0);
        }
```

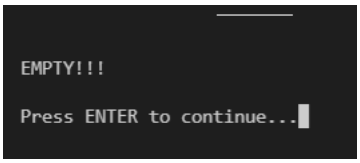
```
    } while(opt != 0);
    printf("\e[1;1H\e[2J");
    printf("Thankyou");
    return 0;
}
```

ScreenShot Hasil

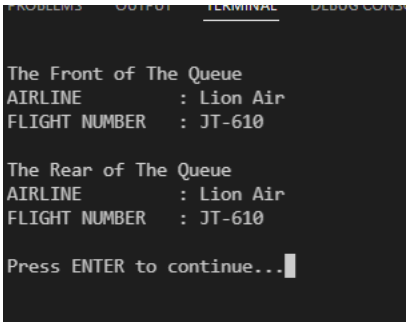




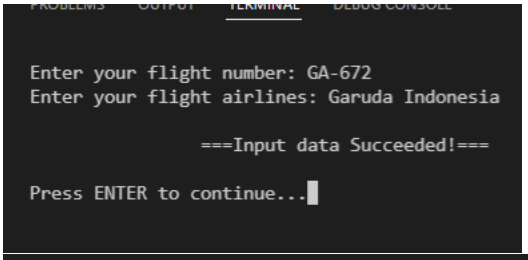
Menu No. 1 (Tidak ada data)

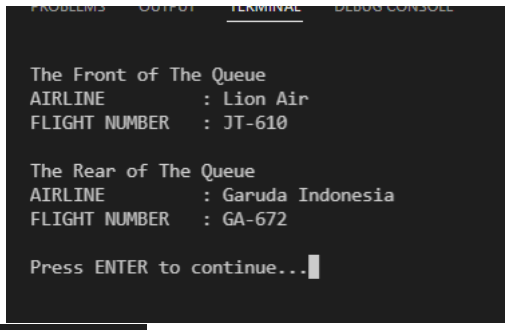


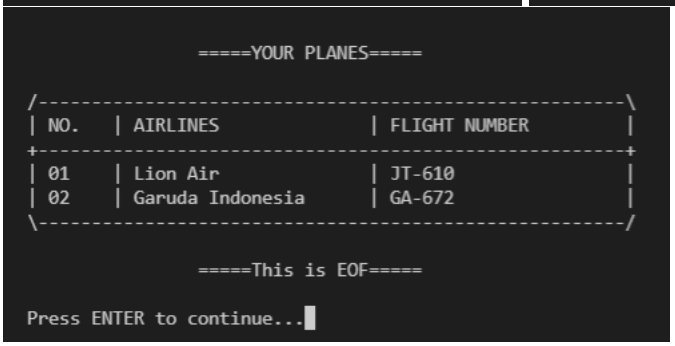
Menu No. 1 (Terdapat data)



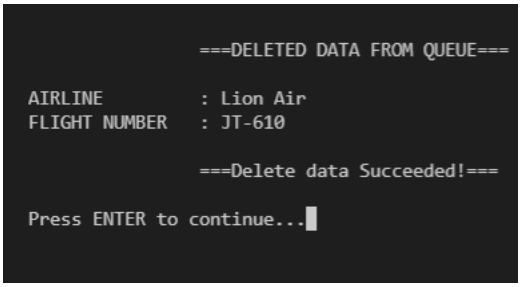
Menu No. 2

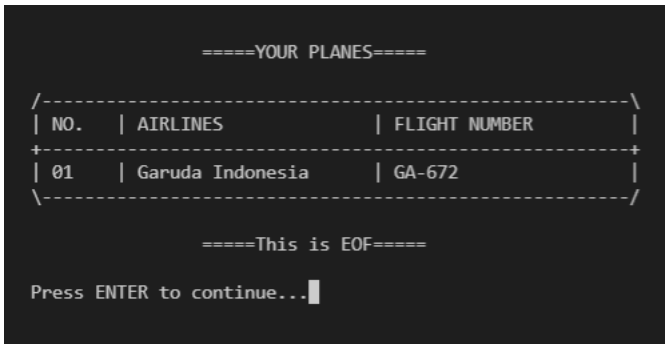






Menu No. 3





Penjelasan Code

Pada bagian fungsi main, user diarahkan untuk memilih ingin menggunakan array atau linked list. Menu didalam dua metode ini sama, yaitu menunjukkan bagian front dan rear dari queue, insert data, deletion, dan display semua data.

1. Array
   1) Queue Status (front_rearArray)
      Fungsi ini berfungsi untuk melihat posisi front dan rear dari queue yang sudah dibuat. Implementasi array yang dapat digunakan dengan cara mengisi MAX menjadi front atau rear.
   2) Add a data (enqueueArray)
      Fungsi ini berfungsi untuk membuat sebuah data baru dengan cara mengisinya melalui rear
   3) Delete a data (dequeArray)
      Deletion dalam array cukup mudah, yaitu dengan “memajukan” front maka secara otomatis rear akan terhapus dan mengganti front serta rear jika tidak terdapat satupun data di dalamnya menjadi -1
   4) Display a data (disArray)
      Mendisplay rangkaian array melalui repetisi for dan akan dilakukan printing sampai dengan rear dari sebuah queue.

2. Linked List
   1) Queue Status (front_rearNode)
      Fungsi ini “sama” dengan implementasi di array yaitu dengan cara mengisi front serta rear pada bagian linked list yang telah dibuat
   2) Add a data (enqueueNode)
      Fungsi ini akan melakukan cek apakah queue tersebut kosong atau tidak. Jika kosong, maka akan dilakukan selection pertama front1 == NULL. Jika terdapat data, maka rear akan diarahkan ke ptr untuk melakukan assigning
   3) Delete a data (dequeNode)
      Fungsi ini memiliki konsep yang sama dengan stack hanya saja pada queue mengambil pada bagian front saja, bukan top
   4) Display a data (disNode)
      Fungsi ini juga memiliki konsep sama dengan stack, yaitu dengan cara repetition while sampai dengan ptr bernilai NULL