Modul 6

Code

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

//Note: printf("\e[1;1H\e[2J"); used to clear the console using regex method

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

struct node{
    float minutes;
    char title[100], artist[100];
    struct node *prev, *next;
} *head = NULL, *tail = NULL;

void display_data(){
    struct node *ptr;
    int num = 0;
    ptr = head;
    if(ptr == NULL){
        printf("EMPTY!!!\n\n");
        printf("Press ENTER to continue insert new data..."); getchar();
        printf("\e[1;1H\e[2J");
    } else{
        printf("/-------------------------------------------------------------------------------
-\\\n");
        printf("| NO.\t| TITLE\t\t\t\t| ARTIST\t\t| DURATION\t|\n");
        printf("+-------------------------------------------------------------------------------
-+\n");
        //Displaying the list
        while(ptr != NULL){
            printf("| %02d\t| %-29s | %-21s | %.2f\t\t|\n", num+1, ptr->title, ptr->artist,
ptr->minutes);
            num++;
            ptr = ptr->next;
        }
        printf("\\-------------------------------------------------------------------------------
--/\n\n");
    }
}

struct node *newNode(){
    struct node *new_node = (struct node*)malloc(sizeof(struct node));
    scanf("%[^\n]", &(new_node->title), printf("Enter the song title: ")); fflush(stdin);
    scanf("%f", &(new_node->minutes), printf("Enter your latest liked songs minute: "));
fflush(stdin);
    scanf("%[^\n]", &(new_node->artist), printf("Enter the song artist: ")); fflush(stdin);
```

```
    new_node->next = NULL;
    new_node->prev = NULL;

    return new_node;
}

void insert_beg(){
    struct node *insert_beg = newNode();

    //Assigning the pointer to the head and clear the previous node to NULL
    if(head == NULL){
        insert_beg->next = NULL;
        insert_beg->prev = NULL;
        head = insert_beg;
        tail = insert_beg;
    } else{
        insert_beg->next = head;
        insert_beg->prev = NULL;
        head->prev = insert_beg;
        head = insert_beg;
    }

    printf("\n\t\t===Input data Succeeded!===\n\n");
    printf("Press ENTER to continue..."); getchar();
    printf("\e[1;1H\e[2J");
}

void insert_end(struct node *insert_end){
    struct node *temp;
    if(head == NULL){
        insert_end->next = NULL;
        insert_end->prev = NULL;
        head = insert_end;
        tail = insert_end;
    } else{
        temp = head;
        while(temp->next != NULL){
            temp = temp->next;
        } temp->next = insert_end;
        insert_end->prev = temp;
        insert_end->next = NULL;
        tail = insert_end;
    }

    printf("\n\t\t===Input data Succeeded!===\n\n");
    printf("Press ENTER to continue..."); getchar();
    printf("\e[1;1H\e[2J");
}

struct node *traverse(float min){
    struct node *queue = head;
    while(queue->minutes != min){
        queue = queue->next;
```

```
        if(queue == NULL){
            break;
        }
    } return queue;
}

void insert_after_given(){
    display_data();
    if(head == NULL){
        insert_beg();
    } else{
        struct node *insert_given, *queue;
        float min;
        scanf("%f", &min, printf("Enter the location (Minute): ")); fflush(stdin);

        queue = traverse(min);

        if(queue != NULL){
            insert_given = newNode();
            if(queue == tail){
                insert_end(insert_given);
            } else{
                insert_given->next = queue->next;
                insert_given->prev = queue;
                queue->next = insert_given;
                queue->next->prev = insert_given;
                printf("\n\t\t===Input data Succeeded!===\n\n");
                printf("Press ENTER to continue..."); getchar();
                printf("\e[1;1H\e[2J");
            }
        } else{
            printf("There's no data before\n\n");
            printf("Press ENTER to continue..."); getchar();
            printf("\e[1;1H\e[2J");
        }
    }
}

void delete_beg(){
    struct node *ptr = head;
    if(ptr == NULL){
        printf("UNDERFLOW!!!\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    } else if(ptr->next == NULL){
        head = NULL;
        free(head);
        printf("\n\t\t===Delete data Succeeded!===\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    } else{
        head = head->next;
        head->prev = NULL;
        free(ptr);
```

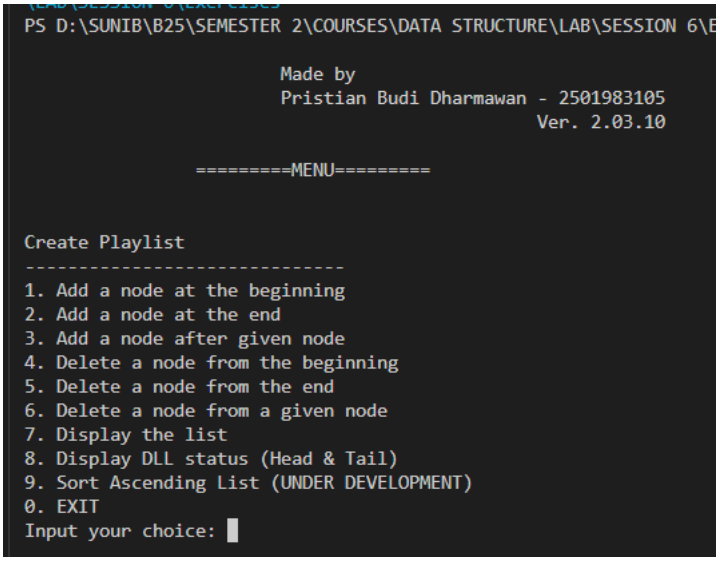
```
        printf("\n\t\t===Delete data Succeeded!===\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    }
}

void delete_end(){
    struct node *ptr = tail;
    if(head == NULL){
        printf("UNDERFLOW!!!\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    } else{
        if(ptr->prev != NULL){
            tail = ptr->prev;
            tail->next = NULL;
        } else{
            head = NULL;
        } free(ptr);
        printf("\n\t\t===Delete data Succeeded!===\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    }
}

void delete_given(){
    display_data();
    if(head == NULL){
        printf("UNDERFLOW!!!\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    } else{
        struct node *insert_given, *queue;
        float min;
        scanf("%f", &min, printf("Enter the location (Minute): ")); fflush(stdin);

        queue = traverse(min);

        if(queue != NULL){
            if(queue == head){
                delete_beg();
            } else if(queue == tail){
                delete_end();
            } else{
                queue->prev->next = queue->next;
                queue->next->prev = queue->prev;
                free(queue);
                printf("\n\t\t===Delete data Succeeded!===\n\n");
                printf("Press ENTER to continue..."); getchar();
                printf("\e[1;1H\e[2J");
            }
        } else{
            printf("There's no data before\n\n");
            printf("Press ENTER to continue..."); getchar();
```

```
            printf("\e[1;1H\e[2J");
        }
    }
}

void display_all(){
    struct node *ptr;
    int num = 0;
    ptr = head;
    if(ptr == NULL){
        printf("EMPTY!!!\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    } else{
        printf("\t\t\t\t=====LIKED SONGS=====\n\n");
        printf("/-------------------------------------------------------------------------------
-\\\n");
        printf("| NO.\t| TITLE\t\t\t\t| ARTIST\t\t| DURATION\t|\n");
        printf("+-------------------------------------------------------------------------------
-+\n");
        printf("\t\t\t    ========HEAD========\n");
        //Displaying the list
        while(ptr != NULL){
            printf("| %02d\t| %-29s | %-21s | %.2f\t\t|\n", num+1, ptr->title, ptr->artist,
ptr->minutes);
            num++;
            ptr = ptr->next;
        }
        printf("\t\t\t    ========TAIL========\n");
        printf("\\-------------------------------------------------------------------------------
--/\n");
        printf("\n\t\t\t\t=====This is EOF=====\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    }
}

void dll_status(){
    struct node *ptr = head, *ptr1 = tail;
    if(ptr == NULL){
        printf("EMPTY!!!\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    } else{
        printf("The Current Head\n");
        printf("TITLE\t\t: %s\n", ptr->title);
        printf("ARTIST\t\t: %s\n", ptr->artist);
        printf("DURATION\t: %.2f\n\n", ptr->minutes);

        printf("The Current Tail\n");
        printf("TITLE\t\t: %s\n", ptr1->title);
        printf("ARTIST\t\t: %s\n", ptr1->artist);
        printf("DURATION\t: %.2f\n\n", ptr1->minutes);
        printf("Press ENTER to continue..."); getchar();
```

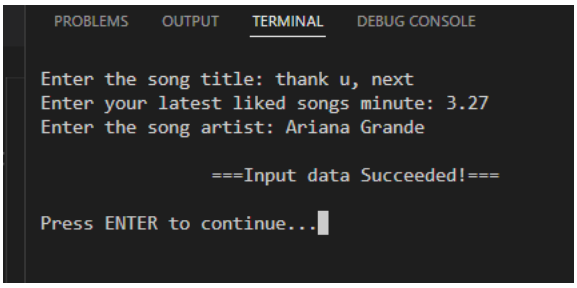
```
        printf("\e[1;1H\e[2J");
    }
}

//UNDER DEVELOPMENT
void dll_sort(){
    /*struct node *curr = NULL, *index = NULL, *temp;
    if(head == NULL){
        printf("EMPTY!!!\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    } else{
        for(curr = head; curr->next != NULL; curr = curr->next){
            for(index = curr->next; index != NULL; index = index->next){
                if(curr->minutes > index->minutes){
                    temp = curr;
                    curr = index;
                    index = temp;
                }
            }
        }
    }*/
}

int main(){
    int opt;
    printf("\n");
    do{
        header();
        printf("\n\n\t\t=========MENU=========\n\n\n");
        printf("Create Playlist\n");
        printf("------------------------------\n");
        printf("1. Add a node at the beginning\n"); //Requirement 1
        printf("2. Add a node at the end\n"); //Requirement 2
        printf("3. Add a node after given node\n"); //Extra Insertion
        printf("4. Delete a node from the beginning\n"); //Requirement 3
        printf("5. Delete a node from the end\n"); //Requirement 4
        printf("6. Delete a node from a given node\n"); //Extra Deletion
        printf("7. Display the list\n"); //Requirement 5
        printf("8. Display DLL status (Head & Tail)\n"); //Requirement 6
        printf("9. Sort Ascending List (UNDER DEVELOPMENT)\n"); //Extra Sorting Mechanism
        printf("0. EXIT\n");
        scanf("%d", &opt, printf("Input your choice: ")); fflush(stdin);
        switch (opt){
        case 1:
            printf("\e[1;1H\e[2J");
            insert_beg(); break;
        case 2:
            printf("\e[1;1H\e[2J");
            insert_end(newNode()); break;
        case 3:
            printf("\e[1;1H\e[2J");
            insert_after_given(); break;
        case 4:
```

```
            printf("\e[1;1H\e[2J");
            delete_beg(); break;
        case 5:
            printf("\e[1;1H\e[2J");
            delete_end(); break;
        case 6:
            printf("\e[1;1H\e[2J");
            delete_given(); break;
        case 7:
            printf("\e[1;1H\e[2J");
            display_all(); break;
        case 8:
            printf("\e[1;1H\e[2J");
            dll_status(); break;
        case 9:
            printf("\e[1;1H\e[2J");
            dll_sort();
            printf("UNDER DEVELOPMENT\n\n");
            printf("Press ENTER to continue..."); getchar();
            break;
        case 0:
            break;
        default:
            printf("\e[1;1H\e[2J");
            printf("There's no menu no %d\n\n", opt);
            printf("Press ENTER to continue..."); getchar();
            printf("\e[1;1H\e[2J");
            break;
        }
    } while(opt != 0);

    printf("\e[1;1H\e[2J");
    printf("Thankyou");

    return 0;
}
```

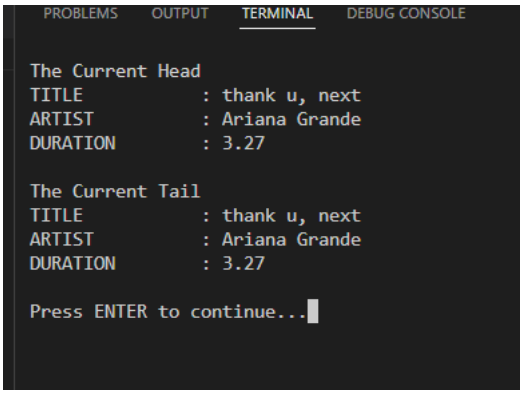
ScreenShot Hasil



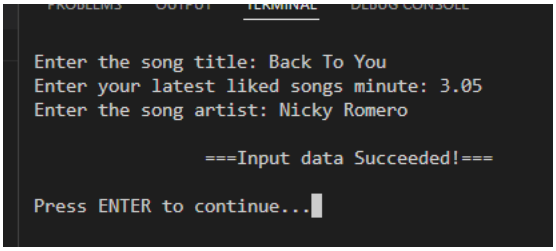
Menu 1



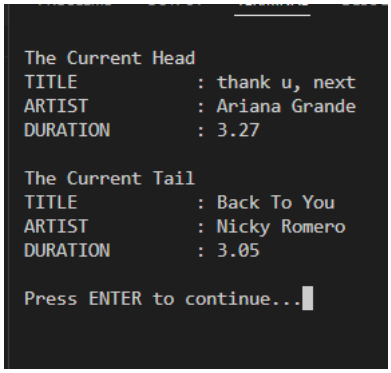
Display (7) & Status (8)



Menu 2



Display (7) & Status (8)



Menu 3

```
/-------------------------------------------------------------------------------\
| NO.   | TITLE                         | ARTIST                | DURATION      |
+-------------------------------------------------------------------------------+
| 01    | thank u, next                 | Ariana Grande         | 3.27          |
| 02    | Back To You                   | Nicky Romero          | 3.05          |
\-------------------------------------------------------------------------------/

Enter the location (Minute): 3.27
Enter the song title: lost
Enter your latest liked songs minute: 2.57
Enter the song artist: Loote

               ===Input data Succeeded!===

Press ENTER to continue...
```

Display (7) & Status (8)

```
                              =====LIKED SONGS=====

/-------------------------------------------------------------------------------\
| NO.   | TITLE                         | ARTIST                | DURATION      |
+-------------------------------------------------------------------------------+
                            ========HEAD========
| 01    | thank u, next                 | Ariana Grande         | 3.27          |
| 02    | lost                          | Loote                 | 2.57          |
| 03    | Back To You                   | Nicky Romero          | 3.05          |
                            ========TAIL========
\-------------------------------------------------------------------------------/

                              =====This is EOF=====

Press ENTER to continue...
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

The Current Head
TITLE           : thank u, next
ARTIST          : Ariana Grande
DURATION        : 3.27

The Current Tail
TITLE           : Back To You
ARTIST          : Nicky Romero
DURATION        : 3.05

Press ENTER to continue...
```

Menu 4

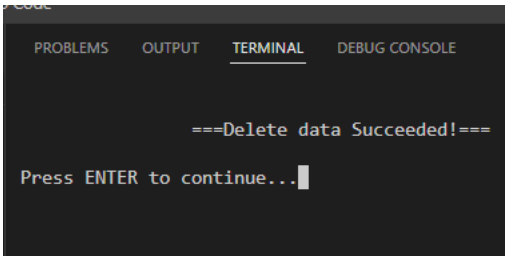
```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

               ===Delete data Succeeded!===

Press ENTER to continue...
```

Display (7) & Status (8)

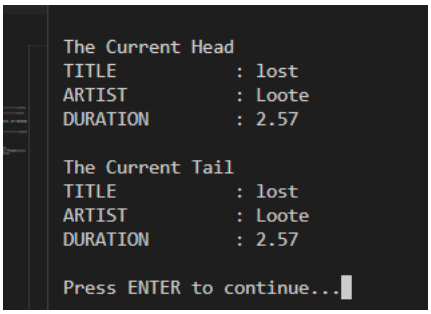
```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  C/C++ Compile Run

                              =====LIKED SONGS=====

/-------------------------------------------------------------------------------\
| NO.   | TITLE                         | ARTIST                | DURATION      |
+-------------------------------------------------------------------------------+
                            ========HEAD========
| 01    | lost                          | Loote                 | 2.57          |
| 02    | Back To You                   | Nicky Romero          | 3.05          |
                            ========TAIL========
\-------------------------------------------------------------------------------/

                              =====This is EOF=====

Press ENTER to continue...
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG

The Current Head
TITLE           : lost
ARTIST          : Loote
DURATION        : 2.57

The Current Tail
TITLE           : Back To You
ARTIST          : Nicky Romero
DURATION        : 3.05

Press ENTER to continue...
```

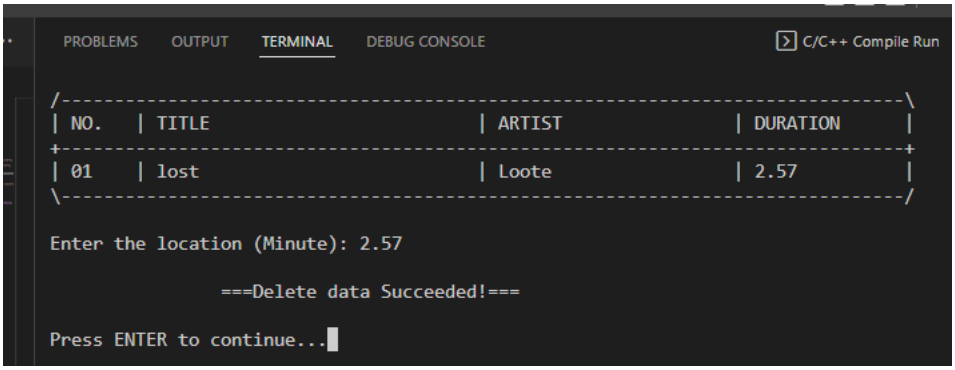
Menu 5



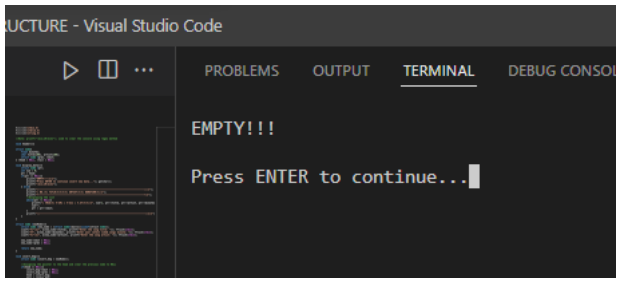
Display (7) & Status (8)



Menu 6



Display (7) & Status (8)



Penjelasan Code

Dalam program ini saya menggunakan 14 Function dan sebuah struct node. Program ini dibuat untuk mendaftarkan lagu yang ingin dimasukkan ke dalam playlist user. User dapat memilih ingin melakukan insertion di awal, akhir ataupun setelah posisi yang diinginkan. Kemudian user juga dapat melakukan deletion pada posisi awal, akhir dan posisi yang diinginkan.

1. Header
   Fungsi ini sebagai penanda bahwa program ini telah dibuat oleh saya sendiri dan terdapat NIM serta version dari program ini

2. Display_data
   Fungsi ini berfungsi untuk menampilkan semua data yang ada di dalam list agar dapat dipasangkan dengan fungsi insert_after_given dan delete_given. Sehingga user tidak perlu mengingat menit yang ia ingin hapus atau ditambahkan. Dengan algorithm ptr yang di assign sebagai head program akan melakukan validasi apakah head = NULL atau tidak. Jika tidak maka akan melakukan print sampai nilai ptr = NULL dan jika iya maka akan melakukan print EMPTY
3. newNode
   Fungsi ini berfungsi untuk menambahkan sebuah data baru ke dalam fungsi insert_beg; insert_end; insert_after_given sehingga saya tidak perlu repot-repot mengetik kembali apa saja yang perlu dimasukkan ke dalam tiga fungsi diatas
4. Insert_beg
   Fungsi ini akan melakukan pengecekan apabila head bernilai NULL maka head dan tail juga memiliki nilai yang sama dari apa yang kita inputkan sedangkan jika head tidak bernilai NULL maka node baru akan diarahkan ke head dan previous akan diassign sebagai NULL sehingga mereplace data sebelumnya yang telah menjadi head.
5. Insert_end
   Fungsi ini sama validasinya dengan fungsi insert_beg hanya saja pada bagian head tidak bernilai NULL maka node temp akan digeser sampai pada bagian akhir list tersebut yaitu next = NULL setelah digeser node baru akan diassign sebagai tail dari list itu tadi
6. Traverse
   Traverse akan menggeser nilai node baru ke arah node yang ingin dilakukan insertion setelahnya
7. Insert_after_given
   Fungsi ini akan melakukan validasi jika head bernilai NULL maka akan memanggil fungsi insert_beg sedangkan jika yang diinputkan bernilai sama dengan tail maka fungsi akan memanggil fungsi insert_end sebelum itu kita harus melakukan traverse atau penggeseran sampai nilai menit yang diinputkan bernilai sama dengan data menit dalam node yaitu dengan pemanggilan fungsi traverse. Kemudian program akan melakukan insertion setelah node lama atau node yang bernilai sama dengan inputan location
8. Delete_beg
   Fungsi ini akan melakukan pengecekan apakah node tersebut bernilai NULL pada bagian previous atau tidak jika ya maka akan dilakukan deletion, jika tidak atau ptr bernilai NULL maka fungsi akan melakukan print UNDERFLOW
9. Delete_end
   Fungsi ini akan mengarahkan ptr sebagai tail dan akan menggeser ptr sampai nilai node terakhir yang bernilai NULL pada bagian next. Setelah itu nilai ptr akan didelete
10. Delete_given
    Fungsi ini sama dengan fungsi insert_after_given hanya saja menukar node setelah yang diinputkan dengan node yang diinputkan sehingga proses deletion akan mendelete nilai yang sama bukan nilai setelah yang sama
11. Display_all
    Fungsi ini akan menampilkan semua list dari data yang telah diinputkan
12. Dll_status
    Fungsi ini menggunakan 2 ptr dimana mengarahkan ke head dan tail. Hal ini digunakan untuk menampilkan status nilai dari list yang ada pada bagian head dan tail nya
13. Dll_sort
    Untuk sementara fungsi ini masih dalam pengembangan karena saya belum menemukan algoritma yang pas untuk diimplementasikan ke dalamnya
14. Main
    Fungsi ini berfungsi untuk mengatur segala fungsi dan menampilkan menu utama dari program ini