Modul 4

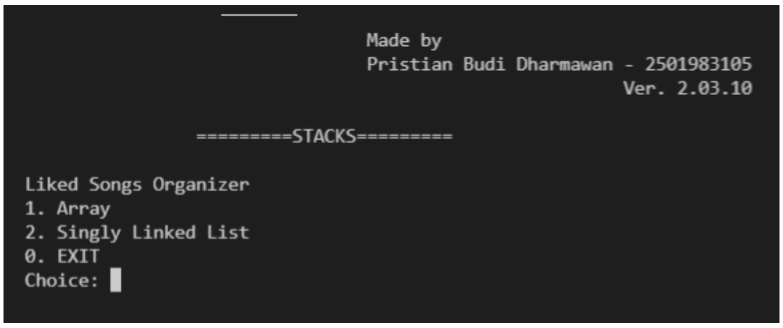
Code

```
#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\t\tVer. 2.03.10");
}

int top = -1;
float liked_min[MAX], min;
char liked_song[MAX][100], liked_artist[MAX][100], song[100], artist[100];

struct linkedList{
    float minutes;
    char title[100], artist2[100];
    struct linkedList *next;
};

struct linkedList *top1 = NULL;

void peekArray(){
    if(top == -1){
        printf("EMPTY!!!\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } else{
        //Print the current top
        printf("The Top of The Stack\n");
        printf("TITLE\t\t: %s\n", liked_song[top]);
        printf("ARTIST\t\t: %s\n", liked_artist[top]);
        printf("DURATION\t: %.2f\n\n", liked_min[top]);
        printf("Press ENTER to continue..."); getchar(); getchar();
    }
}

void pushArray(){
    //Input array to the stack
    scanf("%f", &min, printf("Enter your latest liked songs minute: ")); fflush(stdin);
    scanf("%[^\n]", &song, printf("Enter the song title: ")); fflush(stdin);
    scanf("%[^\n]", &artist, printf("Enter the song artist: ")); fflush(stdin);

    if(top == MAX-1){
        printf("\n\nOVERFLOW!!!\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } else{
        //Filling the top
        top++;
        liked_min[top] = min;
        strcpy(liked_song[top], song);
        strcpy(liked_artist[top], artist);
        printf("\n\t\t===Input data Succeeded!===\n\n");
        printf("Press ENTER to continue..."); getchar();
    }
}

void popArray(){
    if(top == -1){
        printf("UNDERFLOW!!!\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } else{
        //Deleting top of the stacks
        min = liked_min[top];
        strcpy(song, liked_song[top]);
        strcpy(artist, liked_artist[top]);
        top--;

        //Additional information
        printf("\t\t===POPPED DATA FROM STACKS===\n\n");
        printf("TITLE\t\t: %s\n", song);
        printf("ARTIST\t\t: %s\n", artist);
        printf("DURATION\t: %.2f\n\n", min);
        printf("\t\t===Delete data Succeeded!===\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    }
}

void disArray(){
    int num = 0;
    if(top == -1){
        printf("EMPTY!!!\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } 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");

        //Displaying the list
        for(int i=top; i >= 0; i--){
            printf("| %02d\t| %-29s | %-21s | %.2f\t\t|\n", num+1, liked_song[i],
liked_artist[i], liked_min[i]);
            num++;
        }

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

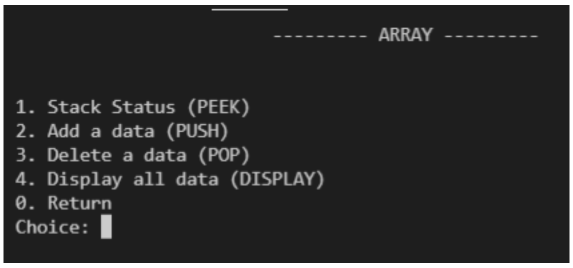
```c
}

void peekNode(){
    if(top1 == NULL){
        printf("EMPTY!!!\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } else{
        //Print the current top
        printf("The Top of The Stack\n");
        printf("TITLE\t\t: %s\n", top1->title);
        printf("ARTIST\t\t: %s\n", top1->artist2);
        printf("DURATION\t: %.2f\n\n", top1->minutes);
        printf("Press ENTER to continue..."); getchar(); getchar();
    }
}

void pushNode(){
    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();
    } else{
        //Input node to the stack
        scanf("%f", &min, printf("Enter your latest liked songs minute: ")); fflush(stdin);
        scanf("%[^\n]", &song, printf("Enter the song title: ")); fflush(stdin);
        scanf("%[^\n]", &artist, printf("Enter the song artist: ")); fflush(stdin);

        if(top1 == NULL){
            ptr->minutes = min;
            *ptr->title = *strcpy(ptr->title, song);
            *ptr->artist2 = *strcpy(ptr->artist2, artist);
            ptr->next = NULL;
            top1 = ptr;
        } else{
            ptr->minutes = min;
            *ptr->title = *strcpy(ptr->title, song);
            *ptr->artist2 = *strcpy(ptr->artist2, artist);
            ptr->next = top1;
            top1 = ptr;
        }

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

void popNode(){
    struct linkedList *ptr = top1;
    if(top1 == NULL){
        printf("UNDERFLOW!!!\n\n");
        printf("Press ENTER to continue..."); getchar();
    } else{
        top1 = top1->next;
        min = ptr->minutes;
        strcpy(song, ptr->title);
        strcpy(artist, ptr->artist2);
        free(ptr);

        //Additional information
        printf("\t\t===POPPED DATA FROM STACKS===\n\n");
        printf("TITLE\t\t: %s\n", song);
        printf("ARTIST\t\t: %s\n", artist);
        printf("DURATION\t: %.2f\n\n", min);
        printf("\t\t===Delete data Succeeded!===\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    }
}

void disNode(){
    struct linkedList *ptr = top1;
    int num = 0;
    if(ptr == NULL){
        printf("EMPTY!!!\n\n");
        printf("Press ENTER to continue..."); getchar(); getchar();
    } 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");

        //Displaying the list
        while(ptr != NULL){
            printf("| %02d\t| %-29s | %-21s | %.2f\t\t|\n", num+1, ptr->title, ptr->artist2,
ptr->minutes);
            num++;
            ptr = ptr->next;
        }

        printf("\\-----------------------------------------------------------------------------
--/\n");
        printf("\n\t\t\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=========STACKS=========\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. Stack Status (PEEK)\n");
                printf("2. Add a data (PUSH)\n");
                printf("3. Delete a data (POP)\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");
                    peekArray(); break;
                case 2:
                    printf("\e[1;1H\e[2J"); printf("\n");
                    pushArray(); break;
                case 3:
                    printf("\e[1;1H\e[2J"); printf("\n");
                    popArray(); 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. Stack Status (PEEK)\n");
                printf("2. Add a data (PUSH)\n");
                printf("3. Delete a data (POP)\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");
                    peekNode(); break;
                case 2:
                    printf("\e[1;1H\e[2J"); printf("\n");
                    pushNode(); break;
                case 3:
                    printf("\e[1;1H\e[2J"); printf("\n");
                    popNode(); 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");
```

```c
    printf("Thankyou");
    return 0;
}
```

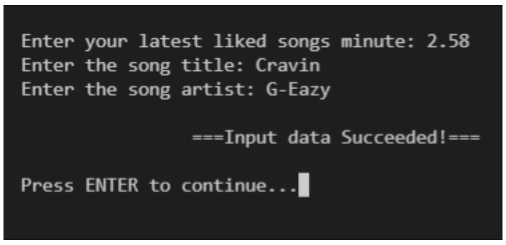
ScreenShot Hasil



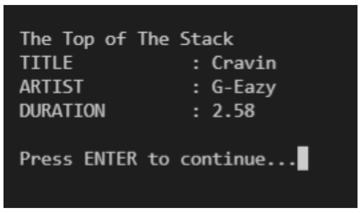
```
                                   Made by
                                   Pristian Budi Dharmawan - 2501983105
                                                            Ver. 2.03.10

                    =========STACKS=========

Liked Songs Organizer
1. Array
2. Singly Linked List
0. EXIT
Choice:
```



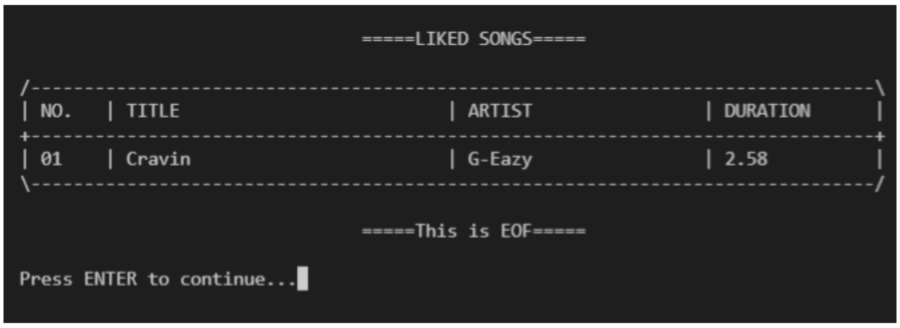
```
                         --------- ARRAY ---------


1. Stack Status (PEEK)
2. Add a data (PUSH)
3. Delete a data (POP)
4. Display all data (DISPLAY)
0. Return
Choice:
```



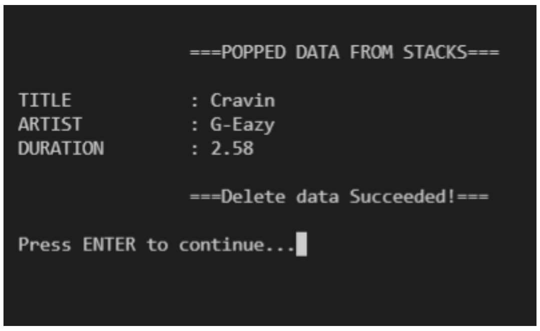
```
Enter your latest liked songs minute: 2.58
Enter the song title: Cravin
Enter the song artist: G-Eazy

                ===Input data Succeeded!===

Press ENTER to continue...
```



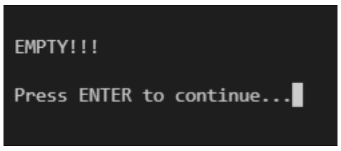
```
The Top of The Stack
TITLE           : Cravin
ARTIST          : G-Eazy
DURATION        : 2.58

Press ENTER to continue...
```



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

/--------------------------------------------------------------------------------\
| NO.   | TITLE                          | ARTIST                 | DURATION      |
+--------------------------------------------------------------------------------+
| 01    | Cravin                         | G-Eazy                 | 2.58          |
\--------------------------------------------------------------------------------/

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

Press ENTER to continue...
```



```
                ===POPPED DATA FROM STACKS===

TITLE           : Cravin
ARTIST          : G-Eazy
DURATION        : 2.58

                ===Delete data Succeeded!===

Press ENTER to continue...
```



```
EMPTY!!!

Press ENTER to continue...
```

Penjelasan Code

Dalam program ini, saya membuat 2 jenis implementasi stacks yang dapat dipilih melalui 2 menu diatas, yaitu array dan single linked list. Setelah memilih salah satu maka user dapat melakukan push, pop, peek, dan display stacks tersebut. Pada dasarnya tampilan kedua menu diatas sama, namun memiliki perbedaan pada implementasi codenya.

Seperti yang diketahui, cara implementasi array sangat mudah karena hanya menggunakan sebuah variabel top untuk membantu "isi" dari sebuah array. Sedangkan linked list sedikit lebih kompleks. Namun, perlu diingat bahwa linked list dalam implementasi array berarti kita hanya menggunakan insertion at the head, dan deletion at the head. Setelah itu kita dapat memodifikasi head apakah mau ditampilkan semua atau hanya bagian head saja. Karena konsep linked list pada algoritma ini cukup dirubah sedikit saja dan kita sudah bisa mendapatkan sebuah stacks yang diinginkan.