```c
/*
Obligatorisk oppgave 1

Oppgave: Generer program som skriver info om bøker

Fremgang min måte:
1.Lage nødvendige globale variabler og funksjoner
2.Lage nødvendige metoder/funksjoner. Komutativt for bruker
3.Kjøre program og sørg for ønsket utskrift
4.Notere logikk og forklaring der det er nødvendig
5.Optimaliser og husk minnefrigjøring!
*/

//*****************************************************//

//Interne includes
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


//Eksterne includes

//*****************************************************//


//Definisjoner
#define STRLEN_MAX 25
#define ANTALL_MAX 100

enum HANDLING
{
    NY,
    LES,
    AVSLUTT
};
//*****************************************************//

//Structs
struct Bok
{
    char mTittel[STRLEN_MAX];
    char mSjanger[STRLEN_MAX];
    int mSider;
};

//*****************************************************//

// Forward deklarasjoner
void Init();
void legg_til_bok();
void print_bøker();
void fri_minne();
```

```c
//*****************************************************//

//Globale variabler
struct Bok** Bibliotek = NULL; //Deklarerer min peker til ett array av Bok-pekere
int antall_boker = 0;

int main()
{
    printf("_______Init: Oblig1_______\n\n");
    Init();
    return 0;
}

void Init()
{
    /*
       Init sørger for at terminalen gir riktig alternativer.
       Tar dermed å kjører valgt alternativ.

    */

    enum HANDLING tilstand = 0;
    printf("Handlinger:\n");
    printf("0 = legg til ny bok.\n");
    printf("1 = printer alle bokene.\n");
    printf("2 = avslutt.\n");
    printf("Velg Handling:");
    scanf_s("%d", &tilstand);
    while (getchar() != '\n'); //Tømmer buffer
    printf("\n");
    switch (tilstand)
    {
    case NY:
    {
        legg_til_bok();
        break;
    }
    case LES:
    {
        print_bøker();
        break;
    }
    case AVSLUTT:
    {
        fri_minne();
        exit(0);
        break;
    }
    default:
    {
        printf("Velg ett gyldig alternativ:\n");
        Init();
```

```
106         printf("\n");
107         break;
108     }
109     }
110     tilstand = 0;
111 }
112
113 void legg_til_bok()
114 {
115     // Lokale variabler for parameter verdier til bok
116     char tittel[STRLEN_MAX] = "\0";
117     char sjanger[STRLEN_MAX] = "\0";
118     int sider = 0;
119
120     //Ber om at bruker skriver inn info
121     printf("[LOG]:Legger til Bok\n-----------------\n");
122     printf("Legg til tittel:");
123     gets_s(tittel, (unsigned)_countof(tittel));
124     printf("Legg til sjanger:");
125     scanf_s("%s", sjanger, (unsigned)_countof(sjanger));
126     printf("Legg til antall sider:");
127     scanf_s("%d",  &sider);
128     while (getchar() != '\n'); //Tømmer buffer
129
130     //Ny bok med allokert minne (allokerer minne: gjør klar en peker
        med samme antall bytes som parameteren)
131     struct Bok* ny_bok = malloc(sizeof(struct Bok));
132
133     //Flag for å se om Bok-structen peker til minne på størresle 0
134     if (ny_bok == NULL)
135     {
136         printf("[WARNING]: Feil med minneallokering..\nReturnerer\n");
137         exit(0);
138     }
139
140     //Kopier string fra parameter bok til lokal bok
141     strcpy_s(ny_bok->mTittel,STRLEN_MAX, tittel);
142     strcpy_s(ny_bok->mSjanger,STRLEN_MAX, sjanger);
143     ny_bok->mSider = sider;
144
145     //Bibliotek med oppdatert minne
146     struct Bok** temp_lib = realloc(Bibliotek, sizeof(struct Bok*) *
        (antall_boker + 1));
147     if (temp_lib == NULL)
148     {
149         printf("[WARNING]: Feil med minneallokering..\nReturnerer\
            n");
150         exit(0);
151     }
152
153     Bibliotek = temp_lib; //Oppdaterer arrayet sin størrelse
154     Bibliotek[antall_boker] = ny_bok; //Oppdaterer elementet på
        indeksen
```

```c
    antall_boker++; // Oppdaterer antall bøker

    printf("-----------------\n[LOG]:Bok Lagt til\n\n");
    Init();
}

void print_bøker()
{
    printf("[LOG]:Printer info\n-----------------\n");

    //Flag for å se om arrayet er en null-peker eller ikke inneholder noen bøker
    if (Bibliotek == NULL || antall_boker == 0)
    {
        printf("[WARNING]: Ingen boker lagt til. Legg til boker\n");
        printf("-----------------\n\n");
        Init();
        return;
    }

    //Deklarerer midlertidige variabler
    int totalt_sider_lest = 0;

    //Itterer over alle bøkene
    for (int i = 0; i < antall_boker; ++i)
    {
        totalt_sider_lest += Bibliotek[i]->mSider;
        printf("Navn: \x1b[1m\x1b[4m%s\x1b[0m \nSjanger: \x1b[1m\x1b[4m%s\x1b[0m\n\n", Bibliotek[i]->mTittel, Bibliotek[i]->mSjanger);
    }
    printf("Boker lest: \x1b[1m\x1b[4m%d\x1b[0m \nSider lest totalt: \x1b[1m\x1b[4m%d\x1b[0m\n", antall_boker, totalt_sider_lest);
    printf("-----------------\n[LOG]:Printing ferdig\n\n");

    Init();
}

void fri_minne()
{
    // Frigjør minne: først hver enkelt bok, dermed arrayet av Bok-pekere
    for (int i = 0; i < antall_boker; ++i)
    {
        free(Bibliotek[i]);
    }
    free(Bibliotek);

    Bibliotek = NULL;
}
```