NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Kontinuasjonseksamensoppgave i PROG1001 – Grunnleggende programmering

Faglig kontakt under eksamen:	**Frode Haug**
Tlf:	**950 55 636**
Eksamensdato:	**6.august 2020 - HJEMME**
Eksamenstid (fra-til):	**09:00-13:00 (4 timer)**
Hjelpemiddelkode/Tillatte hjelpemidler:	**F - Alle trykte og skrevne.** (kalkulator er *ikke* tillatt)
Annen informasjon:	
Målform/språk:	**Bokmål**
Antall sider (inkl. forside):	**7**

Informasjon om trykking av eksamensoppgaven

Originalen er:

1-sidig X **2-sidig** ☐

sort/hvit X **farger** ☐

Skal ha flervalgskjema ☐

Kontrollert av:

_____

Dato Sign

NB: Oppgave 1a, 1b og 2 er totalt uavhengige og kan derfor løses separat.

Oppgave 1 (30%)

I hele oppgave 1, være nøye med å :

- skrive *hele* funksjonen, dvs. *både* headingen og *hele* innmaten
- bruke `const` (når dette er aktuelt) ifm. parametre
- kommentere funksjonen (hva den gjør, evt. parametre og evt. returverdi) etter Doxygen

a) Lag funksjonen `int lesSkrivOgReturnerTall()`

Funksjonen ber brukeren skrive inn tre heltall, og leser så inn tre slik verdier. Den skriver deretter tallene ut igjen i baklengs rekkefølge (ift. hvordan de ble innlest), med *ett* blankt tegn mellom hvert tall. Til slutt returnerer den summen av de tre tallene.

b) Lag funksjonen `void skrivBokstaverMedBindestreker(…)`

Funksjonen får en `char`-array og antall tegn (`int n`) i arrayen som parameter. Bokstavene ligger i indeksene 0 (null) til `n`-1 i arrayen. **Funksjonen går gjennom alle bokstavene, skriver en og en ut på skjermen, med en '-' (bindestrek) mellom hver.** Det skal *ikke* være '-' etter den aller siste bokstaven. Helt til slutt skrives ett linjeskift.
NB: Det ligger altså *ikke* en '\0' helt til slutt i arrayen.

Oppgave 2 (70%)

Les *hele* teksten for denne oppgaven (2a-2g) *nøye*, før du begynner å besvare noe som helst. Studér vedlegget (på utdelte papirer), som inneholder mange viktige opplysninger som du trenger/skal bruke. Legg spesielt merke til `#define`/`const`, structen med datamedlemmer, funksjoner, globale variable, `main()` og ferdiglagd funksjon. Husk også på de ferdiglagde funksjonene for å lese inn data på `LesData.h`. Bruk *alt* dette *svært* aktivt.

Det skal holdes orden på hvilke episoder en person har sett i de ulike sesongene av en lang, lang TV-/nett-serie.

Datastrukturen

Datastrukturen består (se utdelt vedlegg) av arrayen `gSesonger`. I denne er indeksene fra 0 til `gAntallSesonger-1` i bruk. Vedlegget angir også hvilke datamedlemmer `Sesong` inneholder. Hver eneste sesong har en `tittel`, som helt eller delvis kan være lik tittelen på andre sesonger. `antallEpisoder` er antallet ulike episoder (1-20) i *en* sesong. Dette kan variere noe fra sesong til sesong. Denne variabelen er altså *ikke* antall episoder brukeren har sett. Hvilke episoder som er sett (av de `antallEpisoder` mulige), ligger lagret som `true`/`false` i `sett`.
NB: Sesongene og episodene angis for/av brukeren (både ved utskrift og innlesning) fra 1 til `gAntallSesonger`/`antallEpisoder`, selv om de i arrayene ligger lagret fra indeks nr.0 til `gAntallSesonger`-1/`antallEpisoder`-1.
*Vedlegget inneholder alt du trenger av structer, datamedlemmer og globale variable for å løse denne eksamensoppgaven. Dessuten er **prototyper** for funksjoner stort sett også deklarert/definert.*

Oppgaven

a) **Skriv innmaten til** `void nySesong()` **og**
`void sesongLesData(struct Sesong* s)`

Den første funksjonen kommer med en melding om det ikke er plass til flere sesonger. I motsatt fall skrives den nye sesongens nummer ut på skjermen, og en ny `Sesong` opprettes. Den andre funksjonen leser inn dennes to første datamedlemmer, og setter *alle aktuelle* elementer i `sett` til `false`. (Innlesing av hvilke episoder som er sett, skjer i oppgave 2C.)

b) **Skriv innmaten til** `void skrivAlleSesongene()` **og**
`void sesongSkrivData(const struct Sesong* s)`

Den første funksjonen sørger for at *alle* sesongene får skrevet ut sitt nummer/indeks, og vha. den andre funksjonen blir *alle* data inni hver `sesong` skrevet ut. Episodene skal angis med numre, og om sett ('X') eller ei ('-'), med følgende eksempel på utseende:

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
 X  X  -  -  X  -  -  X  -  X  X  X  X  -  -  -
```

c) **Skriv innmaten til** `void settEnEpisode()` **og**
`void sesongSettEpisode(struct Sesong* s)`

Den første funksjonen spør om et aktuelt sesongnummer. Deretter registreres det *en* ny sett episode denne sesongen vha. den andre funksjonen. Som altså spør om et lovlig episodenummer. Er denne episoden *ikke* sett ennå, registreres den som det, og det kommer en kvitteringsmelding. I motsatt fall kommer det en melding om at den er allerede (registrert som) sett.

d) **Skriv innmaten til** `void skrivEnNavngittSesong()`
og all annen kode/funksjoner som trengs for å skrive ut indeksen til og *alle* dataene om «alle» de sesongene som i tittelen *inneholder* en tekst innskrevet av brukeren. Ble ingen sesong skrevet/ funnet, kommer det en egen melding om dette. **NB:** *Hele* tittelen trenger *ikke* å være identisk. Det holder at *ett eller annet sted* i tittelen så forekommer *hele* den innskrevne teksten.
Hint: Bruk bl.a. biblioteksfunksjonen `strstr(…)`

e) **Skriv innmaten til** `void sesongerProsentvisSett()`
og all annen kode/funksjoner som trengs for å skrive ut *hver* sesong sin indeks, sammen med antall prosent (som heltall) av sesongens episoder som er sett. Til slutt skriver den ut nummeret og prosenten for den sesongen det er sett flest prosentvis episoder i. Er flere like, skrives bare den første ut.

f) **Skriv innmaten til** `void skrivTilFil()` **og**
`void sesongSkrivTilFil(FILE* ut, const struct Sesong* s)`
Funksjonene sørger til sammen for at *hele* datastrukturen blir skrevet til «SESONGER.DTA». `gAntallSesonger` *skal* ligge aller først på filen.
Filformatet ellers bestemmer du helt selv, men ***skal*** **oppgis i besvarelsen**.

g) **Skriv innmaten til** `void lesFraFil()` **og**
`void sesongLesFraFil(FILE* inn, struct Sesong* s)`
Funksjonene sørger til sammen for at *hele* datastrukturen blir lest inn fra «SESONGER.DTA», etter det formatet du selv var med på å bestemme i forrige deloppgave.

Annet (klargjørende):

- Episodene er hver eneste sesong er innholdsmessig såpass uavhengige (jfr. «Klovn», «Side om side», «Helt perfekt» og «Neste sommer»). Derfor trenger ikke brukeren hverken å ha sett episoder og sesonger sekvensielt og fullstendig.
- Gjør dine egne forutsetninger og presiseringer av oppgaven, dersom du skulle finne dette nødvendig. Gjør i så fall klart rede for disse *i starten* av din besvarelse av oppgaven.
- **NB:** Det skal *ikke* brukes C++-kode, dvs. slikt som f.eks: string-klassen, kode fra STL, templates eller andre større hjelpebiblioteker. Men, de vanligste includer brukt i hele høst er tilgjengelig.

Lykke til med alle serier som må/skal/kan/bør følges videre i livet!

FrodeH

Vedlegg til PROG1001, 6.aug. 2020: Halvferdig programkode

```
#include <stdio.h>              //  printf, scanf, FILE
#include <stdlib.h>             //  sizeof, malloc, free
#include <string.h>             //  strcpy, strlen, strstr
#include <stdbool.h>            //  bool, true, false
#include "LesData.h"            //  Verktøykasse for lesing av diverse data


#define   MAXSESONG    100      ///<  Max. antall sesonger.
#define   MAXEPISODE    20      ///<  Max. antall episoder EN sesong.
const int STRLEN = 60;          ///<  Max. tekstlengde.


/**
 *  Sesong (med tittel, antall episoder og hvilke som er sett).
 */
struct Sesong {
    char* tittel;               //  Sesongens tittel/emne.
    int   antallEpisoder;       //  Antall episoder i vedkommende sesong.
    bool  sett[MAXEPISODE];     //  Sett episode eller ei.
};


void skrivMeny();
void nySesong();                                            //  Oppgave 2A
void sesongLesData(struct Sesong* s);                       //  Oppgave 2A
void skrivAlleSesongene();                                  //  Oppgave 2B
void sesongSkrivData(const struct Sesong* s);               //  Oppgave 2B
void settEnEpisode();                                       //  Oppgave 2C
void sesongSettEpisode(struct Sesong* s);                   //  Oppgave 2C
void skrivEnNavngittSesong();                               //  Oppgave 2D
void sesongerProsentvisSett();                              //  Oppgave 2E
void skrivTilFil();                                         //  Oppgave 2F
void sesongSkrivTilFil(FILE* ut, const struct Sesong* s);   //  Oppgave 2F
void lesFraFil();                                           //  Oppgave 2G
void sesongLesFraFil(FILE* inn, struct Sesong* s);          //  Oppgave 2G


int    gAntallSesonger;               ///<  Antall sesonger hittil registrert.
struct Sesong* gSesonger[MAXSESONG];  ///<  Alle sesongene.


/**
 *  Hovedprogrammet:
 */
int main()  {
    char kommando;

    lesFraFil();                                            //  Oppgave 2G
    skrivMeny();
    kommando = lesChar("Ønske");

    while (kommando != 'Q') {
        switch(kommando) {
           case 'N': nySesong();                    break;  //  Oppgave 2A
           case 'A': skrivAlleSesongene();          break;  //  Oppgave 2B
           case 'E': settEnEpisode();               break;  //  Oppgave 2C
           case 'S': skrivEnNavngittSesong();       break;  //  Oppgave 2D
           case 'P': sesongerProsentvisSett();      break;  //  Oppgave 2E
           default:  skrivMeny();                   break;
        }
        kommando = lesChar("Ønske");
    }

    skrivTilFil();                                          //  Oppgave 2F
```

```
    printf("\n\n");
    return 0;
}


/**
 *  Skriver/presenterer programmets muligheter/valg for brukeren.
 */
void skrivMeny() {
    printf("\nFØLGENDE KOMMANDOER ER LOVLIG:\n");
    printf("\tN   = Ny sesong\n");
    printf("\tA   = skriv Alle sesongene\n");
    printf("\tE   = sett en ny Episode\n");
    printf("\tS   = skriv en navngitt Sesong\n");
    printf("\tP   = sesonger Prosentvis sett\n");
    printf("\tQ   = Quit/avslutt\n");
}



/**
 *  Oppgave 2A - Legger inn (om mulig) en ny sesong i datastrukturen.
 *
 *  @see   sesongLesData(...)
 */
void nySesong() {                                   /*   LAG INNMATEN   */   }


/**
 *  Oppgave 2A - Leser inn relevante data og nullstiller andre.
 *
 *  @param  s -  Sesongen som får innlest/initiert sine data
 */
void sesongLesData(struct Sesong* s) {              /*   LAG INNMATEN   */   }



/**
 *  Oppgave 2B - Skriver ALT om ALLE sesonger.
 *
 *  @see   sesongSkrivData(...)
 */
void skrivAlleSesongene() {                         /*   LAG INNMATEN   */   }


/**
 *  Oppgave 2B - Skriver ALT om EN sesong ut på skjermen.
 *
 *  @param  s - Sesongen som skrives ut
 */
void sesongSkrivData(const struct Sesong* s) {      /*   LAG INNMATEN   */   }



/**
 *  Oppgave 2C - Sett EN episode en gitt sesong.
 *
 *  @see   sesongSettEpisode(...)
 */
void settEnEpisode() {                              /*   LAG INNMATEN   */   }
```

```
/**
 *  Oppgave 2C - Registrerer (om mulig) at EN episode er sett.
 *
 *  @param  s - Sesongen det er sett EN ny episode
 */
void sesongSettEpisode(struct Sesong* s) {           /*   LAG INNMATEN   */   }


/**
 *  Oppgave 2D - Skriver navngitt(e) sesong(er).
 */
void skrivEnNavngittSesong() {   /*  LAG INNMATEN OG ANNEN KODE/FUNKSJON(ER)  */  }


/**
 *  Oppgave 2E - Skriver % sett av hver sesong, og den mest sette sesongen.
 */
void sesongerProsentvisSett() {  /*  LAG INNMATEN OG ANNEN KODE/FUNKSJON(ER)  */  }


/**
 *  Oppgave 2F - Skriver ALLE sesongene til fil.
 *
 *  @see   sesongSkrivTilFil(...)
 */
void skrivTilFil() {                                 /*   LAG INNMATEN   */   }

/**
 *  Oppgave 2F - Alle EN sesongs data skrives ut på fil.
 *
 *  @param   ut  - Filen det skal skrives til
 *  @param   s   - Sesongen som skrives til fil
 */
void sesongSkrivTilFil(FILE* ut, const struct Sesong* s) {
                                                     /*   LAG INNMATEN   */   }


/**
 *  Oppgave 2G - Leser ALLE sesongene fra fil.
 *
 *  @see     sesongLesFraFil(...)
 */
void lesFraFil() {                                   /*   LAG INNMATEN   */   }

/**
 *  Oppgave 2G - Leser ALT om EN sesong fra fil.
 *
 *  @param  inn - Filen det skal leses fra
 *  @param  s   - Sesongen som får innlest sine data
 */
void sesongLesFraFil(FILE* inn, struct Sesong* s) {  /*   LAG INNMATEN   */   }
```