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:	**9.august 2022**
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):	**9**

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%)

a) Hva blir utskriften fra følgende program (litt hjelp: det blir 5 linjer):

```
#include <stdio.h>
#include <string.h>
char txt[] =
 "ENDELIG-VAR-DEN-TILBAKELAGTE-SOMMEREN-NOENLUNDE-NORMAL-IGJEN";


int main() {
    int i = 31 / 4, j = 17 - 14 % 2,  k;

    k = strlen(txt) / i;
    printf(" %c %c %c\n", txt[k], txt[j], txt[i]);


    for (i = 0;  i < 60;  i += 15)
        printf(" %c", txt[i]);
    printf("\n");


    i = 14;  j = 36;  k = 59;
    while (txt[i] == txt[j]  &&  txt[j] == txt[k])  {
       i--;  --j;  --k;
    }
    printf(" %c %c %c\n", txt[i], txt[j], txt[k]);


    i = 29;  j = i + 2;
    while (txt[i--] != txt[j++])  ;
    printf(" %c %c\n", txt[i], txt[j]);


    i = 34;   j = 1;
    while (txt[i] >= 'R'  ||  txt[i] < 'H')  {
        i -= 5;  j *=2;
    }
    printf(" %c %c\n", txt[i], txt[j]);


    return 0;
}
```

b) **Hva blir utskriften fra følgende program (litt hjelp: det blir 5 linjer):**

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

struct Ferie {
  char hvor[30];
  int  antD, antP;
};


void S22Funk1(const struct Ferie* f)  {
  printf("%s, %i dager, %i personer\n",
                                    f->hvor, f->antD, f->antP);
}


int S22Funk2(const struct Ferie f1, const struct Ferie f2)  {
  return ((f1.hvor[1] > f2.hvor[3]) ? 17 : 1);
}


bool S22Funk3(const struct Ferie* f1, const struct Ferie* f2)  {
  return (strcmp(f1->hvor, f2->hvor) &&  f1->antP == f2->antP);
}


struct Ferie* S22Funk4(const struct Ferie f1)  {
   struct Ferie* f2
                = (struct Ferie*) malloc(sizeof(struct Ferie));
   strcpy(f2->hvor, f1.hvor);
   f2->antD = f1.antD + 2;  f2->antP = 1;
}


int main() {
    struct Ferie  ferie1 = { "Hellas",   21, 2 },
                  ferie2 = { "Kroatia",  14, 8 },
                  ferie3 = { "Stavern",   7, 2 };
    struct Ferie* ferie4;

    S22Funk1(&ferie2);
    printf("%i\n", S22Funk2(ferie1,   ferie2));
    printf("%i\n", S22Funk3(&ferie1, &ferie3));
    ferie4 = S22Funk4(ferie3);   S22Funk1(ferie4);
    printf("%i\n", S22Funk3(S22Funk4(ferie1), &ferie3));

    return 0;
}
```

Oppgave 2 (70%)

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

Det holdes orden på ulike ting/produkt/saker som må handles/er handlet inn ifm oppussing (i et hus/leilighet/hybel).

Datastrukturen

Datastrukturen består (se vedlegget) av arrayen `gTing`. I denne er indeksene fra 0 til `gAntallTing-1` i bruk. Vedlegget angir også hvilke datamedlemmer structen inneholder.
NB: Ved innlesning/utskrift angis tingene via numrene 1 til `gAntallTing`, selv om de i arrayen altså ligger lagret fra indeks nr.0 til `gAntallTing-1`.

*Vedlegget inneholder alt du trenger av structer, datamedlemmer og globale variable for å løse denne eksamensoppgaven. Dessuten er **prototyper** for alle funksjoner også ferdig deklarert/definert.*

Oppgaven

a) **Skriv innmaten til** `void nyTing()` **og**
`void tingLesData(struct Ting* ting)`

Den første funksjonen kommer med en melding om det er fullt med ting. I motsatt fall skriver den ut den nye tingens nummer, en ny ting opprettes/lages og *alle* dens datamedlemmer lese inn vha. den andre funksjonen. `antall` skal være i intervallet 1-1000, mens `pris` skal være i intervallet 0-10000. Til slutt (i den første funksjonen) telles antall ting opp med 1 (en).
Bruk *meget aktivt* ferdiglagde funksjoner (både i vedlegget og fra filen `LesData.h`).

b) **Skriv innmaten til** `void skrivAlleTingene()` **og**
`void tingSkrivData(const struct Ting* ting)`

Den første funksjonen kommer med en melding om ingen ting finnes. I motsatt fall går den gjennom alle registrerte ting. For hver av dem skrives tingens nummer (fra 1 (en) og oppover), samt *alle* tingens data (vha. den andre funksjonen), inkludert at enum-verdier skrives som tekst.

c) **Skriv innmaten til** `void endrePris()` **og**
`void tingEndrePris(struct Ting* ting)`

Den første funksjonen kommer også med en melding om ingen ting finnes. I motsatt fall leser den en tings nummer, og brukeren får anledning til å endre/sette denne tingens pris (via den andre funksjonen, som også både i starten og til slutt skriver tingens aktuelle pris).

d) **Skriv innmaten til** `void skrivAlleMedTekst()` **og**
`bool tingMedTekst(const struct Ting* ting, const char tekst[])`
Den første funksjonen kommer også med en melding om ingen ting finnes. I motsatt fall leser den en ønsket tekst. Deretter gås det igjennom alle tingene, og de som *inneholder* den ønskede teksten (som *hele eller del av* sitt `navn`. Den andre funksjonen sørger for å finne ut dette) får sitt nummer og *alle* data skrevet ut på skjermen.

e) **Skriv innmaten til** `void totalKostnad()` **og**
`float tingHentKostnad(const struct Ting* ting)`
Den første funksjonen går igjennom alle tingene, og finner ut *totalkostnadene* for alle dem som får fått en pris. Dvs. *totalsummen* av *alle* tingenes `pris` * `antall`. Dette aller siste regnes *pr.ting* ut av den andre funksjonen. Den første funksjonen skriver ut totalsummen, antall ulike ting som inngår i denne summen, samt antall ting som hittil *ikke* har fått noen pris (for der vil jo dens kostnaden være lik 0 (null), da `pris` er 0).

f) **Skriv innmaten til** `void skrivTilFil()` **og**
`void tingSkrivTilFil(FILE* ut, const struct Ting* ting)`
Funksjonene sørger til sammen for at *alle* tingene blir skrevet til filen «TING.DTA». `gAntallTing` *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 tingLesFraFil(FILE* inn, struct Ting* ting)`
Funksjonene sørger til sammen for at *alle* tingene blir lest inn fra filen «TING.DTA», etter det formatet du selv bestemte i forrige deloppgave.

Annet (klargjørende):

- 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 oppussingen av ett eller annet sted …..!

FrodeH

Vedlegg til PROG1001, august 2022: Halvferdig programkode

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


#define   MAXTING        100    ///<  Max. antall ting/produkt som trengs.
const int STRLEN  =       80;   ///<  Max. tekstlengde.


/**
 *  Enhet (ugyldig/udefinert, centimeter, meter, kvadratmeter, stykk, liter).
 */
enum Enhet { ugyldig, cm, m, kvm, stk, l};


/**
 *  Ting (med navn, antall, pris og enhetsmål).
 */
struct Ting {
  char* navn;                   //  Tingens:  - navn
  float antall,                 //            - antallet av tingen
        pris;                   //            - pris pr.enhet
  enum  Enhet enhet;            //            - enhetsmål
};


char  charFraEnum(const enum Enhet enhet);                    //  |
enum  Enhet enumFraChar(const char tegn);                     //  | Ferdig-
char  lesEnhetsBokstav();                                     //  | laget.
void  skrivMeny();                                            //  |
void  nyTing();                                               //  Oppgave 2A
void  tingLesData(struct Ting* ting);                         //  Oppgave 2A
void  skrivAlleTingene();                                     //  Oppgave 2B
void  tingSkrivData(const struct Ting* ting);                 //  Oppgave 2B
void  endrePris();                                            //  Oppgave 2C
void  tingEndrePris(struct Ting* ting);                       //  Oppgave 2C
void  skrivAlleMedTekst();                                    //  Oppgave 2D
bool tingMedTekst(const struct Ting* ting, const char tekst[]);//  Oppgave 2D
void  totalKostnad();                                         //  Oppgave 2E
float tingHentKostnad(const struct Ting* ting);               //  Oppgave 2E
void  skrivTilFil();                                          //  Oppgave 2F
void  tingSkrivTilFil(FILE* ut, const struct Ting* ting);     //  Oppgave 2F
void  lesFraFil();                                            //  Oppgave 2G
void  tingLesFraFil(FILE* inn, struct Ting* ting);            //  Oppgave 2G


int    gAntallTing;                    ///<  Antall ting hittil registrert.
struct Ting* gTing[MAXTING];           ///<  Alle tingene.


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

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

    while (kommando != 'Q') {
```

```
        switch (kommando) {
          case 'N':  nyTing();                      break;    //  Oppgave 2A
          case 'A':  skrivAlleTingene();            break;    //  Oppgave 2B
          case 'E':  endrePris();                   break;    //  Oppgave 2C
          case 'I':  skrivAlleMedTekst();           break;    //  Oppgave 2D
          case 'T':  totalKostnad();                break;    //  Oppgave 2E
          default:   skrivMeny();                   break;
        }
        kommando = lesChar("\nØnske");
    }

    skrivTilFil();                                            //  Oppgave 2F
    return 0;
}


/**
 *  Gjør om en lovlig enum-verdi til ett tegn/bokstav.
 *
 *  @param    enhet  -  Enum-verdi som skal konverteres til bokstav/tegn
 *  @return   En bokstav (C, M, K, S, L) ut fra en enum-verdi
 */
char charFraEnum(const enum Enhet enhet)  {
  switch (enhet)  {                       //  Ut fra enum-verdi returneres
    case  cm:   return 'C';               //    aktuell bokstav:
    case   m:   return 'M';
    case kvm:   return 'K';
    case stk:   return 'S';
    case   l:   return 'L';
  }
}


/**
 *  Gjør om en (lovlig) bokstav til en aktuell enum-verdi.
 *
 *  @param    tegn  -  Bokstav/tegn som skal konverteres til enum-verdi
 *  @return   Enum-verdi ut fra en bokstav/tegn
 */
enum Enhet enumFraChar(const char tegn)  {
  switch (tegn)  {
    case 'C':  return  cm;                //  Ut fra bokstav returneres
    case 'M':  return   m;                //    aktuell enum-verdi:
    case 'K':  return kvm;
    case 'S':  return stk;
    case 'L':  return   l;
    default:   printf("\n\tUgyldig bokstav for Enhet!\n\n");
               return ugyldig;
  }
}


/**
 *  Leser korrekt en lovlig bokstav for en Enhet, og returnerer denne.
 *
 *  @return   En bokstav (C, M, K, S, L) som står for en enhetsbetegnelse
 */
char lesEnhetsBokstav()  {
  char tegn;
  do  {                                       //  Sikrer LOVLIG bokstav-verdi:
    tegn = lesChar("\tEnhetstype (C(m), M, K(vm), S(tk), L)");
  } while (tegn != 'C' && tegn != 'M' &&
           tegn != 'K' && tegn != 'S' && tegn != 'L');
  return tegn;
}
```

```
/**
 *  Skriver/presenterer programmets muligheter/valg for brukeren.
 */
void skrivMeny() {
  printf("\nFØLGENDE KOMMANDOER ER LOVLIG:\n");
  printf("\tN   = Ny ting som skal kjøpes\n");
  printf("\tA   = skriv Alle tingene som er kjøpt/skal kjøpes\n");
  printf("\tE   = Endre/legge inn prisen for en ting\n");
  printf("\tI   = skriv alle tingene som Inneholder en tekst i navnet\n");
  printf("\tT   = Totalkostnad (så langt) for alt handlet/oppussingen\n");
  printf("\tQ   = Quit/avslutt\n");
}


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


/**
 *  Oppgave 2A - Leser inn ALLE datamedlemmer i EN ting.
 *
 *  @param   ting -  Tingen som får innlest sine data
 *  @see     enumFraChar(...)
 */
void  tingLesData(struct Ting* ting)  {             /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2B - Skriver ALT om ALLE tingene.
 *
 *  @see   tingSkrivData(...)
 */
void  skrivAlleTingene()  {                         /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2B - Skriver ALT om EN ting ut på skjermen.
 *
 *  @param   ting  -  Tingen som skrives ut
 */
void  tingSkrivData(const struct Ting* ting)  {     /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2C - Endre en tings pris.
 *
 *  @see   tingEndrePris(...)
 */
void  endrePris()  {                                /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2C - Leser inn EN tings nye pris.
 *
 *  @param    ting  -  Tingen som får endret sin pris
 */
void  tingEndrePris(struct Ting* ting)  {           /*   LAG INNMATEN   */  }
```

```
/**
 *  Oppgave 2D - Skriver ALLE tingene som har en gitt (sub)tekst i 'navn'.
 *
 *  @see   tingMedTekst(...)
 *  @see   tingSkrivData(...)
 */
void  skrivAlleMedTekst()  {                          /*   LAG INNMATEN   */  }

/**
 *  Oppgave 2D - Returnerer om en ting har en gitt tekst i 'navn' eller ei.
 *
 *  @param    ting   -  Tingen som skal sjekkes for om inneholder 'tekst'
 *  @param    tekst  -  Teksten det sjekkes for om er i tingens 'navn'
 *  @return   Om tingens 'navn' inneholder 'tekst' eller ei
 */
bool tingMedTekst(const struct Ting* ting, const char tekst[])  {
                                                      /*   LAG INNMATEN   */  }

/**
 *  Oppgave 2E - Beregner og skriver TOTALkostand for ALLE tingene HITTIL.
 *
 *  @see   tingHentKostnad((...)
 */
void  totalKostnad()  {                               /*   LAG INNMATEN   */  }

/**
 *  Oppgave 2E - Regner ut og returnerer EN tings TOTALE kostnad/pris.
 *
 *  @param    ting  -  Tingen som får sin totalkostnad beregnet
 *  @return   Tingens totale kostnad
 */
float tingHentKostnad(const struct Ting* ting)  {
                                                      /*   LAG INNMATEN   */  }

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

/**
 *  Oppgave 2F - Alle EN tings data skrives ut på fil.
 *
 *  @param   ut    -  Filen det skal skrives til
 *  @param   ting  -  Tingen som skrives til fil
 *  @see     charFraEnum(...)
 */
 void  tingSkrivTilFil(FILE* ut, const struct Ting* ting)  {
                                                      /*   LAG INNMATEN   */  }

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

/**
 *  Oppgave 2G - Leser ALT om EN ting fra fil.
 *
 *  @param   inn   -  Filen det skal leses fra
 *  @param   ting  -  Tingen som får innlest sine data
 *  @see     enumFraChar(...)
 */
void  tingLesFraFil(FILE* inn, struct Ting* ting)  { /*   LAG INNMATEN   */  }
```