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:	**10.august 2023**
Eksamenstid (fra-til):	**09:00-13:00 (4 timer)**
Hjelpemiddelkode/Tillatte hjelpemidler:	**I - Alle trykte og skrevne.** (kalkulator er *ikke* tillatt)
Annen informasjon:	
Målform/språk:	**Bokmål**
Antall sider (inkl. forside):	**8**

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>

char txt[] = "NAA-ER-ENDELIG-SOMMEREN-SLUTT-OG-STUDIENE-KAN-STARTE-IGJEN";

int main() {
    int i = 18, j = i % 5, k = i / j;
    do {
        printf("%c %c\n", txt[i], txt[i + j]);
        i += k;   j *= 2;
    } while (i <= 30);

    i = j = k = 12 - 3 * 2;
    while (k  <  i * j) {
        printf("%c %c\n", txt[k], txt[k * 5 / 3]);
        k += 15;
    }
    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 Hall {
    char hall[30], by[30];
    int  tilskuere;
};

void HallFunk1(const struct Hall h)  {
    printf("%s: %s (%i)", h.hall, h.by, h.tilskuere);  }

bool HallFunk2(const struct Hall* h1, const struct Hall* h2)  {
    return (h1->tilskuere  <  h2->tilskuere);  }

bool HallFunk3(const struct Hall* h, const char* t)  {
    return (strstr(h->by, t));  }

void HallFunk4(struct Hall* h, const char* t)  {
    strcat(h->by, t);  }

struct Hall HallFunk5(const struct Hall h1, const struct Hall* h2) {
    struct Hall h;
    strcpy(h.hall, h1.hall);   strcpy(h.by, h2->by);
    h.tilskuere = h1.tilskuere + h2->tilskuere;
    return h;  }

int main() {
    struct Hall hall1 = { "Telenor Arena", "Oslo", 25000 },
                hall2 = { "Oslo Spektrum", "Oslo", 10500 },
                hall3 = { "Koengen", "Bergen", 24000 };
    HallFunk1(hall1);   printf("   ");   HallFunk1(hall3);   printf("\n");
    if (HallFunk2(&hall2, &hall3)) printf("Banan\n"); else  printf("Eple\n");
    printf("%i %i\n", HallFunk3(&hall1, "Oslo"), HallFunk3(&hall3, "Oslo"));
    HallFunk4(&hall3, "/Festning");    HallFunk1(hall3);     printf("\n");
    hall2 = HallFunk5(hall3, &hall1);  HallFunk1(hall2);     printf("\n");
    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`, structene 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 snøbrøytere/sandstrøere og deres kunder en vintersesong.

Datastrukturen

Datastrukturen består (se vedlegget) av arrayene `gBroytereStroere` og `gKunder`.
I disse er henholdsvis indeksene fra 0 til `gAntallBroytStro-1`/`gAntallKunder-1` i bruk.
Vedlegget angir også hvilke datamedlemmer structene inneholder.
NB: Ved innlesning/utskrift angis brøytere/strøere og kunder via numrene 1 og oppover, selv om de i arrayene altså ligger lagret fra indeks nr.0 og oppover.

*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 skrivAlleKunder()` **og**
`void kundeSkrivData(const struct Kunde* kunde)`

Den første funksjonen kommer med en melding om ingen kunder finnes. I motsatt fall går den gjennom alle registrerte kunder. For hver av dem skrives kundens nummer (fra 1 (en) og oppover), samt *alle* kundens data (vha. den andre funksjonen). I stedet for kundens `prisNr` skrives aktuell sum fra `PRIS`-arrayen. Og i stedet for `BSNr` skrives aktuell brøyters navn vha. ferdiglaget funksjon.

b)

Skriv innmaten til `void nyKunde()` **og**
`void kundeLesData(struct Kunde* kunde)`

Den første funksjonen kommer med en melding om det er fullt med kunder. I motsatt fall skriver den ut den nye kundens nummer, en ny kunde opprettes/lages og dens fem første datamedlemmer lese inn vha. den andre funksjonen. De to siste nullstilles. Husk at `prisNr` og `BSNr` lagres som *reelle* indeks-verdier fra 0 (null) og oppover. Til slutt (i den første funksjonen) telles antall kunder opp med 1 (en). Bruk *meget aktivt* ferdiglagde funksjoner på filen `LesData.h`.

c)

Skriv innmaten til `void broytetStroddHosKunder()` **og**
`void kundeBroytetStrodd(struct Kunde* kunde)`

Den første funksjonen leser kundenumre inntil 0 (skrives). Det blir så vha. den andre funksjonen registrert *en* brøyting eller strøing hos vedkommende kunde. Den andre funksjonen *sikrer* at kun B(røyting) eller S(trøing) skrives, og aktuelt datamedlem telles opp.

(Du trenger *ikke* å sjekke at det virkelig finnes både kunder og brøytere. Dessuten trenger du ikke å sjekke at kunden *virkelig* er hos brøyteren som har aktivert kommandoen 'G'.
Det kan selvsagt skje at samme kundenummer skrives flere ganger, fordi det har forekommet både brøyting og strøing, eller at brøyteren henger etter med registreringen).

d) **Skriv innmaten til** `void lagFakturaer()` **og**
`void kundeLagFaktura(const struct Kunde* kunde)`
Den første funksjonen kommer også med en melding om ingen kunder finnes. I motsatt fall lager/skriver den ut på skjermen fakturaene for *alle* kundene (vha. den andre funksjonen).
Den andre funksjonen sørger for at alle kundens data skrives ut, akkurat som i oppgave 2A, samt kundens totale kostnad/utgift (bruk her bl.a. ferdiglaget funksjon).

e) **Skriv innmaten til** `void endreBroyterStroer()` **og**
`void kundeEndreBroyterStroer(struct Kunde* kunde)`
Den første funksjonen kommer også med en melding om ingen kunder finnes. I motsatt fall spørres om et kundenummer. Er *ikke* denne kundens to datamedlemmer for antall brøytinger/strøinger lik null (bruk også her ferdiglaget funksjon), kommer det en egen melding.
I motsatt fall kalles den andre funksjonen. Denne skriver først nummeret (fra 1 og oppover) og navnet for brøyteren som kunden har. Deretter tilbys det å endre dette til et annet brøyternummer (husk å lagre dette fra 0 (null) og oppover).

f) **Skriv innmaten til** `void broyterStroerMedMestInntjening()`
Funksjonen (og evt. andre funksjoner du måtte finne det hensiktsmessig å lage) skal finne ut hvilken brøyter som har mest utstående hos alle sine kunder. Dvs. hvilken brøyter som vil tjene mest når fakturaer sendes til alle kundene. Til slutt skrives denne brøyterens navn, og den summen kundene til sammen skylder vedkommende. *Er det flere brøytere som har samme sum, skal bare den første skrives ut, men det skal komme en utskrift med at flere har samme sum.*

g) **Skriv innmaten til** `void lesFraFil()` **og**
`void kundeLesFraFil(FILE* inn, struct Kunde* kunde)`
Funksjonene sørger til sammen for at *alle hendelsene* blir lest inn fra filen «KUNDER.DTA», `gAntallKunder` *skal* ligge aller først på filen. **Filformatet** ellers bestemmer du helt selv, men ***skal*** **oppgis i besvarelsen**.

Annet (klargjørende):

- I teksten brukes «brøyter/strøer» synonymt med bare «brøyter» (for korthets skyld).
  Dvs. *alle* brøytere tilbyr også strøing. Og *alle til samme pris for begge deler og for samme areal*.
- Det er ikke kommandoer for f.eks: legge inn ny brøyter (dette forutsetter vi at også leses inn fra fil), skrive ut alle brøytere, slette brøyter/kunde, endre navn/tlf/adresse og `prisNr` hos brøyter/kunde.
- Når en vintersesong er over, regner vi at programmet brukes slik: fakturaer skrives (oppgave 2D), antall brøytinger/strøinger hos hver kunde nullstilles (*ikke* en del av eksamensoppgaven å lage dette). Deretter kan kommandoen 'E' (oppgave 2E) greit kjøres.
- 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.

Antar mange gleder seg til at det snart kommer snø og is som skal brøytes/strøs!

FrodeH

Vedlegg til PROG1001, august 2023: Halvferdig programkode

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


#define   MAXBROYTSTRO  20      ///<  Max. antall brøytere/strøere.
#define   MAXKUNDER    100      ///<  Max. antall kunder.
const int STRLEN  =     80;     ///<  Max. tekstlengde.
const int PRIS[]  = { 175, 325, 475};  ///<  Faste priser for ulike arealer.


/**
 *  Brøyter/strøer (med kun navn og (mobil)telefon).
 */
struct BroyterStroer {        //  Brøyters/strøers:
  char* navn;                 //     - navn
  int   tlf;                  //     - (mobil)telefonnummer
};



/**
 *  Kunde (med navn, adresse, (mobil)telefon, prisgruppe, brøyter/strøer-nr,
 *          antall brøytinger/strøinger hittil i løpet av sesongen).
 */
struct Kunde {                //  Kundens:
  char* navn,                 //     - navn
      * adresse;              //     - adresse
  int   tlf,                  //     - (mobil)telefonnummer
        prisNr,               //     - prisgruppenummer (0-2)
        BSNr,                 //     - brøyter/strøers reelle nummer/indeks
                              //                          (fra 0 og oppover)
        antBroyt,             //     - antall brøytinger (hittil)
        antStro;              //     - antall strøinger (hittil)
};


void skrivMeny();                                                 //  |
int  kundeHentBSNr(const struct Kunde* kunde);                    //  | Ferdig-
int  kundeHentKostnad(const struct Kunde* kunde);                 //  |  -laget.
void broytStroSkrivNavn(const struct BroyterStroer* broytStro);//  |
void skrivAlleKunder();                                           //  Oppgave 2A
void kundeSkrivData(const struct Kunde* kunde);                   //  Oppgave 2A
void nyKunde();                                                   //  Oppgave 2B
void kundeLesData(struct Kunde* kunde);                           //  Oppgave 2B
void broytetStroddHosKunder();                                    //  Oppgave 2C
void kundeBroytetStrodd(struct Kunde* kunde);                     //  Oppgave 2C
void lagFakturaer();                                              //  Oppgave 2D
void kundeLagFaktura(const struct Kunde* kunde);                  //  Oppgave 2D
void endreBroyterStroer();                                        //  Oppgave 2E
void kundeEndreBroyterStroer(struct Kunde* kunde);                //  Oppgave 2E
void broyterStroerMedMestInntjening();                            //  Oppgave 2F
void lesFraFil();                                                 //  Oppgave 2G
void kundeLesFraFil(FILE* inn, struct Kunde* kunde);              //  Oppgave 2G


int    gAntallBroytStro;      ///<  Antall brøytere/strøere hittil registrert.
struct BroyterStroer* gBroytereStroere[MAXBROYTSTRO];  ///<  Brøytere/strøere.
int    gAntallKunder;         ///<  Antall kunder hittil registrert.
struct Kunde* gKunder[MAXKUNDER];                      ///<  Alle kundene.
```

```
/**
 *  Hovedprogrammet.
 */
int main()  {
    char kommando;

    lesFraFil();                                                    //  Oppgave 2G

    skrivMeny();
    kommando = lesChar("\nValg");

    while (kommando != 'Q') {
        switch (kommando) {
          case 'S':  skrivAlleKunder();                 break;  //  Oppgave 2A
          case 'K':  nyKunde();                         break;  //  Oppgave 2B
          case 'G':  broytetStroddHosKunder();          break;  //  Oppgave 2C
          case 'F':  lagFakturaer();                    break;  //  Oppgave 2D
          case 'E':  endreBroyterStroer();              break;  //  Oppgave 2E
          case 'I':  broyterStroerMedMestInntjening(); break;  //  Oppgave 2F
          default:   skrivMeny();                       break;
        }
        kommando = lesChar("\nValg");
    }

    return 0;
}



/**
 *  Skriver/presenterer programmets muligheter/valg for brukeren.
 */
void skrivMeny() {
    printf("\nFØLGENDE KOMMANDOER ER LOVLIG:\n");
    printf("\tS   = Skriv alle kunder\n");
    printf("\tK   = ny Kunde\n");
    printf("\tG   = gjort Gjøremål (brøytet/strødd) hos kunde(r)\n");
    printf("\tF   = lag/skriv Fakturaer\n");
    printf("\tE   = Endre til annen brøyter/strøer for en kunde\n");
    printf("\tI   = finn og skriv brøyter/strøer med mest Inntjening\n");
    printf("\tQ   = Quit/avslutt\n");
}



/**
 *  Retunerer nummeret/indeksen for brøyteren/strøeren til en kunde.
 *
 *  @param    Kunde  -  Aktuell kunde å returnere nummeret for
 *  @return   Brøyterens/strøerens nummer/indeks for en kunde
 */
int kundeHentBSNr(const struct Kunde* kunde)  {
   return (kunde->BSNr);
}



/**
 *   Returnerer kundens totale kostnad/sum for all brøyting/strøing.
 *
 *   @param   kunde  -  Kunden det hentes kostnad for
 */
int kundeHentKostnad(const struct Kunde* kunde)  {
  return ((kunde->antBroyt + kunde->antStro) * PRIS[kunde->prisNr]);
}
```

```
/**
 *  Skriver eb brøyters/strøers navn ut på skjermen.
 *
 *  @param    broytStro  -  Aktuell brøyter/strøer å skrive ut navnet for
 */
void broytStroSkrivNavn(const struct BroyterStroer* broytStro)  {
   printf("%s", broytStro->navn);
}



/**
 *  Oppgave 2A - Skriver ALT om ALLE kunder ut på skjermen.
 *
 *  @see   kundeSkrivData(...)
 */
void skrivAlleKunder()  {                              /*   LAG INNMATEN   */  }



/**
 *  Oppgave 2A - Skriver ALT om EN kunde ut på skjermen.
 *
 *  @param   kunde  -  Kunden som skrives ut
 *  @see     ……
 */
void kundeSkrivData(const struct Kunde* kunde)  {     /*   LAG INNMATEN   */  }



/**
 *  Oppgave 2B - Legger inn (om mulig) en ny kunde inn i datastrukturen.
 *
 *  @see   kundeLesData(...)
 */
void nyKunde()  {                                      /*   LAG INNMATEN   */  }



/**
 *  Oppgave 2B - Leser inn ALLE datamedlemmene i EN kunde.
 *
 *  @param   kunde  -  Kunden som får innlest sine data
 */
void kundeLesData(struct Kunde* kunde)  {              /*   LAG INNMATEN   */  }



/**
 *  Oppgave 2C - Registrerer brøyting/strøing hos kunder.
 *
 *  @see   kundeBroytetStrodd(...)
 */
void broytetStroddHosKunder()  {                       /*   LAG INNMATEN   */  }



/**
 *  Oppgave 2C - Registrerer brøyting/strøing hos EN kunde.
 *
 *  @param    kunde  -  Kunden det registreres brøyting/strøing hos
 */
void kundeBroytetStrodd(struct Kunde* kunde)  {        /*   LAG INNMATEN   */  }
```

```
/**
 *  Oppgave 2D - Skriver/lager (om mulig) fakturaer for ALLE kundene.
 *
 *  @see   kundeLagFaktura(...)
 */
void lagFakturaer()  {                              /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2D - Skriver ut på skjermen faktura (kostnad) for EN kunde.
 *
 *  @param    kunde  -  Kunden det skrives/lages faktura for
 *  @see      ……
 */
void kundeLagFaktura(const struct Kunde* kunde)  {   /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2E - Endrer (om mulig) brøyter/strøer for en kunde.
 *
 *  @see      ……
 */
void endreBroyterStroer()  {                        /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2E - Endrer en kundes brøyter/strøer.
 *
 *  @param    kunde  -  Kundens som får sin brøyter/strøer endret
 *  @see      ……
 */
void kundeEndreBroyterStroer(struct Kunde* kunde)  { /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2F - Finner og skriver brøyter/strøer med mest inntjening.
 *
 *  @see      ……
 */
void broyterStroerMedMestInntjening()  {            /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2G - Leser ALLE kundene (og ALLE brøytere/strøere) inn fra fil.
 *
 *  @see   kundeLesFraFil(...)
 */
void lesFraFil() {                                  /*   LAG INNMATEN   */  }


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