NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamensoppgave i PROG1001
– Grunnleggende programmering

Faglig kontakt under eksamen:	**Frode Haug**
Tlf:	**950 55 636**
Eksamensdato:	**29.november 2021**
Eksamenstid (fra-til):	**15:00-19:00 (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-KAN-VI-HA-FYSISK-UNDERVISNING-PAA-SKOLEN-IGJEN";


int main() {
    int i = 43 / 4, j = 12 + 2 * 2, k;

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


    do  {
      printf(" %c", txt[i]);
    } while (++i < j);
    printf("\n");


    i = 17 / 3;  j = 17 % 6;
    for (k = i;  k <= 32;  k += j)
        printf(" %c", txt[k]);
    printf("\n");


    i = 17;  j = 29;
    while (i < j) {  i++;  j--;  };
    printf(" %c %c\n", txt[i], txt[j]);


    i = 17;  j = 29;
    while (i++ < j--) ;
    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 Dose {
  char hva[30];
  int  dato;
};


void H21Funk1(const struct Dose d)  {
  printf("%s, satt den %i/%i-%i\n", d.hva,
                  d.dato%100,  (d.dato/100)%100,  d.dato/10000);
}


int H21Funk2(const struct Dose* d1, const struct Dose* d2)  {
  return ((d1->dato > d2->dato) ? d2->dato : d1->dato);
}


bool H21Funk3(const struct Dose d1, const struct Dose* d2)  {
  return (d1.hva[4] == d2->hva[5]);
}


int H21Funk4(const struct Dose* d1, const struct Dose* d2)  {
  return( (!strcmp(d1->hva, d2->hva)) ? d1->dato : d2->dato);
}

void H21Funk5(const struct Dose* d1,
              const struct Dose* d2, struct Dose* d3) {
  strcat(d3->hva, d2->hva);   strcat(d3->hva, d1->hva);
}


int main() {
    struct Dose dose1 = { "Pfizer",   210615},
                dose2 = { "Moderna",  210817},
                dose3 = { "Aquavit",  211224};

    H21Funk1(dose2);
    printf("%i\n", H21Funk2(&dose3, &dose1));
    printf("%i\n", H21Funk3(dose2,  &dose1));
    strcpy(dose2.hva, dose1.hva);
    printf("%i\n", H21Funk4(&dose2, &dose1));
    H21Funk5(&dose1, &dose2, &dose3);  H21Funk1(dose3);

    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 også er på utdelte papirer), som inneholder mange viktige opplysninger som du trenger/skal bruke. Legg spesielt merke til `#define`/`const`, structene med datamedlemmer, funksjoner, globale variable, `main()` og *fem 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 kunder, banker og lånene som kundene har i bankene.

Datastrukturen

Datastrukturen består (se vedlegget) av arrayene `gKunder` og `gBanker`. I disse er indeksene fra 0 til `gAntallKunder-1/gAntallBanker-1` i bruk. Vedlegget angir også hvilke datamedlemmer structene inneholder. **NB:** Bankene angis for/av brukeren (ved I/O) fra 1 til `gAntallBanker`, selv om de i arrayen altså ligger lagret fra indeks nr.0 til `gAntallBanker-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 skrivAlleBankene()` **og**
`void bankSkrivData(const struct Bank* bank)`
Den første funksjonen sørger for at *alle* bankene og deres numre (nr.1 →…) blir skrevet ut på skjermen. Vha. den andre funksjonen blir *alle* data inni hver `Bank` skrevet ut.

b) **Skriv innmaten til** `void endreBanksRente()` **og**
`void bankEndreRente(struct Bank* bank)`
Den første funksjonen kommer med en melding om ingen banker finnes. I motsatt fall leses et banknummer (1 til `gAntallBanker`). Deretter endres denne bankens rente vha. den andre funksjonen. Denne funksjonen skriver både aller først og helt til slutt bankens aktuelle standard rente. Mellom dette tilbys brukeren å endre renten til intervallet 0.2 til *max. 2.0 prosentpoeng høyere enn den nåværende renten.*

c) **Skriv innmaten til** `void skrivEnKunde()` **og**
`void kundeSkrivData(const struct Kunde* kunde)`
Den første funksjonen kommer med en melding om ingen kunder finnes. I motsatt fall leses et kundenavn. Det forsøkes å finne denne kunden vha. allerede ferdiglaget funksjon. Er kunden *ikke* å finne, kommer det også en melding. I motsatt fall skrives alle kundens data vha. den andre funksjonen. Denne funksjonen skriver *alle* kundens data, inkludert navnet på den *første* lånebanken (bruk ferdiglaget funksjon), og det *månedlige* kronebeløpet for rentene (= lån * (rente/100) / 12) i denne banken. Renten hentes vha. ferdiglaget funksjon. *Bankens navn og månedlige kronebeløp skrives også om kunden har et lån i bank nr.2.*

d) ***Angi*** **innmaten til** `void endreKundesLaan()` **og**
skriv innmaten til `void kundeEndreLaan(struct Kunde* kunde)`
Den første funksjonen gjør akkurat det samme som den første i oppgave 2C, bare at `kundeEndreLaan(...)` kalles i stedet. **Forklar kun hva endringen i koden da vil bli.**
Den andre funksjonen skriver både aller først og helt til slutt alle kundens data. Mellom dette spør den om hvilket lån som skal endres (nr.1 eller 2). Lånet settes til en verdi mellom 0 og `MAXLAAN`. Er lånet helt nytt (lånebank er hittil nr.0 (null)), så leses og legges også inn et lovlig banknummer (1 til `gAntallBanker`).
NB: Men du trenger *ikke* å sjekke at de to lånene er i ulike banker. Skulle lånet bli satt til 0 (null), så fortsetter kunden allikevel å ligge der, og med den samme banken som lånegiver.

e) **Skriv innmaten til** `void bankMedFlestKunder()` **og**
`bool kundeLaanIBank(const struct Kunde* kunde, const int nr)`
Den andre funksjonen returnerer `true/false` til om kunden har lån i bank nummer `nr`.
Den første funksjonen finner ut hvilken bank som har flest lånekunder. Den skriver ut dette antallet og bankens navn. Har flere banker dette samme antallet kunder, så skrives den første banken ut. Ble ingen bank funnet med mer enn null lånekunder, så kommer en egen melding.

f) **Skriv innmaten til** `void skrivTilFil()` **og**
`void kundeSkrivTilFil(FILE* ut, const struct Kunde* kunde)`
Funksjonene sørger til sammen for at *alle kundene* blir skrevet til filen «KUNDER.DTA». `gAntallKunder` *skal* ligge aller først på filen. **Filformatet** ellers bestemmer du helt selv, men ***skal*** **oppgis i besvarelsen**. (Bankene skal altså *ikke* skrives til fil.)

g) **Skriv innmaten til** `void lesFraFil()` **og**
`void kundeLesFraFil(FILE* inn, struct Kunde* kunde)`
Funksjonene sørger til sammen for at *alle kundene* blir lest inn fra filen «KUNDER.DTA», etter det formatet du selv bestemte i forrige deloppgave. (Bankene skal *ikke* leses fra fil.)

Annet (klargjørende):

- Bankene er altså identifisert vha. sitt nummer (1 til `gAntallBanker`), mens kundene via sitt navn (som vi forutsetter alle er unike/ulike).
- Etter at det er lest fra fil, forutsetter vi at *alle* kundene har *minst ett* lån (kroner null eller mer) i en bank allerede, dvs. at `laan[0]` og `bankNr[0]` er fylt med reelle verdier hos alle kundene.
- *Alle* kunder i *samme* bank har (noe urealistisk) samme standard rente i denne banken.
- **Det skal altså *ikke* lages funksjonalitet for bl.a å:**
  - skrive ut *alle* kundene (*kun* for å skrive alle bankene, jfr. oppg.2A)
  - **legge inn en ny eller slette en kunde/bank**
  - endre banken for et allerede eksisterende lån hos en kunde (også når det er null)
  - endre navn, adresse, mail og telefon inni structene
- Programmet har ikke noe om automatisk reduksjon av lånet (avskrivning) pr.mnd.
- 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 livets små og store (studie-/bolig-/bil-)lån! FrodeH

Vedlegg til PROG1001, november 2021: Halvferdig programkode

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


#define   MAXKUNDER      500     ///<  Max. antall kunder.
#define   MAXBANKER       20     ///<  Max. antall banker.
const int STRLEN  =       80;    ///<  Max. tekstlengde.
const int MAXLAAN = 30000000;    ///<  Max. lån (30 millioner).

/**
 *  Kunde (med navn, mailadresse, mobilnr og lånene i max.to ulike banker).
 */
struct Kunde {
  char* navn,                    //  Kundens:  - navn
      * mail;                    //            - mail-adresse
  int   tlf;                     //            - mobilnummer
  float laan[2];                 //            - lånenes størrelse
  int   bankNr[2];               //            - indeksen på lånebankene
};


/**
 *  Bank (med navn, adresse, mail, telefon og nåværende rente).
 */
struct Bank  {
   char* navn,                   //  Bankens:  - navn
       * adresse,                //            - gate- og stedsadresse
       * mail;                   //            - mail-adresse
   int   tlf;                    //            - telefonnummer
   float stdRente;               //            - nåværende standard rente
};


void  skrivMeny();                                                  //  |
float bankHentRente(const struct Bank* bank);                       //  | Ferdig-
void  bankSkrivNavn(const struct Bank* bank);                       //  | laget:
int   finnKunde(const char* nvn);                                   //  |
bool  kundeHeter(const struct Kunde* kunde, const char* nvn);       //  |
void  skrivAlleBankene();                                           //  Oppgave 2A
void  bankSkrivData(const struct Bank* bank);                       //  Oppgave 2A
void  endreBanksRente();                                            //  Oppgave 2B
void  bankEndreRente(struct Bank* bank);                            //  Oppgave 2B
void  skrivEnKunde();                                               //  Oppgave 2C
void  kundeSkrivData(const struct Kunde* kunde);                    //  Oppgave 2C
void  endreKundesLaan();                                            //  Oppgave 2D
void  kundeEndreLaan(struct Kunde* kunde);                          //  Oppgave 2D
void  bankMedFlestKunder();                                         //  Oppgave 2E
bool  kundeLaanIBank(const struct Kunde* kunde, const int nr);      //  Oppgave 2E
void  skrivTilFil();                                                //  Oppgave 2F
void  kundeSkrivTilFil(FILE* ut, const struct Kunde* kunde);        //  Oppgave 2F
void  lesFraFil();                                                  //  Oppgave 2G
void  kundeLesFraFil(FILE* inn, struct Kunde* kunde);               //  Oppgave 2G


int    gAntallKunder;                  ///<  Antall kunder hittil registrert.
struct Kunde* gKunder[MAXKUNDER];      ///<  Alle kundene.
int    gAntallBanker;                  ///<  Antall banker hittil registrert.
struct Bank*  gBanker[MAXBANKER];      ///<  Alle bankene.
```

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

    lesFraFil();                                                    //  Oppgave 2G

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

    while (kommando != 'Q') {
        switch (kommando) {
          case 'B':   skrivAlleBankene();          break;    //  Oppgave 2A
          case 'R':   endreBanksRente();           break;    //  Oppgave 2B
          case 'K':   skrivEnKunde();              break;    //  Oppgave 2C
          case 'E':   endreKundesLaan();           break;    //  Oppgave 2D
          case 'F':   bankMedFlestKunder();        break;    //  Oppgave 2E
          default:    skrivMeny();                 break;
        }
        kommando = lesChar("\nØnske");
    }

    skrivTilFil();                                                  //  Oppgave 2F

    return 0;
}


/**
 *  Skriver/presenterer programmets muligheter/valg for brukeren.
 */
void skrivMeny() {
    printf("\nFØLGENDE KOMMANDOER ER LOVLIG:\n");
    printf("\tB   = skriv ALLE Bankene\n");
    printf("\tR   = endre en banks standard Rente\n");
    printf("\tK   = skriv EN Kunde\n");
    printf("\tE   = Endre (øke/minske) en kundes lån\n");
    printf("\tF   = skriv banken med Flest kunder\n");
    printf("\tQ   = Quit/avslutt\n");
}


/**
 *  Returnerer en banks for øyeblikket standard rente.
 *
 *  @param    bank  -  Banken som får returnert sin rente
 *  @return   'bank' sin standard rente
 */
float bankHentRente(const struct Bank* bank)  {
  return (bank->stdRente);
}


/**
 *  Skriver KUN en banks navn.
 *
 *  @param   bank  -  Banken som får skrevet ut sitt navn
 */
void bankSkrivNavn(const struct Bank* bank)  {
  printf("%s", bank->navn);
}
```

```
/**
 *  Prøver å finne indeksen i 'gKunder' for en navngitt kunde.
 *
 *  @param   nvn  -  Søkt kundes navn
 *  @return  Indeksen for kunden, evt. -1 om ikke ble funnet
 */
int finnKunde(const char* nvn)  {
    for (int i = 0;  i < gAntallKunder;  i++)
        if (!strcmp(gKunder[i]->navn, nvn))  return i;   //  Funn/match!

    return -1;                           //  Ikke funnet noen med dette 'nvn'.
}


/**
 *  Returnerer om en kunde har et gitt navn eller ei.
 *
 *  @param    kunde  -  Kunden som skal sjekkes om har navnet 'nvn'
 *  @param    nvn    -  Navnet å sjekke om 'kunde' har eller ei
 *  @return   Om kunden har navnet (true) 'nvn' eller ei (false)
 */
bool kundeHeter(const struct Kunde* kunde, const char* nvn) {
  return (!strcmp(kunde->navn, nvn));
}


/**
 *  Oppgave 2A - Skriver ALT om ALLE banker.
 *
 *  @see   bankSkrivData(...)
 */
void skrivAlleBankene() {                                   /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2A - Skriver ALT om EN bank ut på skjermen.
 *
 *  @param   bank  -  Banken som skrives ut
 */
void bankSkrivData(const struct Bank* bank) {               /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2B - Endrer en banks rente.
 *
 *  @see   bankEndreRente(...)
 */
void endreBanksRente() {                                    /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2B - Leser inn en banks nye standard rente.
 *
 *  @param   bank  -  Banken som får endret sin rente
 */
void bankEndreRente(struct Bank* bank) {                    /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2C - Skriver ALLE data om en navngitt kunde.
 *
 *  @see   finnKunde(...)
 *  @see   kundeSkrivData(...)
 */
void skrivEnKunde() {                                       /*  LAG INNMATEN  */  }
```

```
/**
 *  Oppgave 2C - Skriver ALLE data om EN kunde ut på skjermen.
 *  @param   kunde  -  Kunden som får skrevet ut ALLE sine data
 *  @see     bankSkrivNavn(...)
 *  @see     bankHentRente(...)
 */
void kundeSkrivData(const struct Kunde* kunde) {           /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2D - Endrer en kundes lån.
 *  @see   finnKunde(...)
 *  @see   kundeEndreLaan(...)
 */
void endreKundesLaan() {                                   /*  LAG INNMATEN  */  }

/**
 *  Oppgave 2D - Endrer ett av lånene for en kunde.
 *  @param    kunde  -  Kunden som får endret ETT av sine lån
 *  @see      kundeSkrivData(...)
 */
void kundeEndreLaan(struct Kunde* kunde) {                 /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2E - Finner og skriver banken med flest lånekunder.
 *  @see      kundeLaanIBank(...)
 *  @see      bankSkrivNavn(...)
 */
void bankMedFlestKunder() {                                /*  LAG INNMATEN  */  }

/**
 *  Oppgave 2E - Finner ut om kunden har lån i bank nummer 'nr'.
 *  @param    kunde  -  Aktuell kunde å sjekke for lånebank
 *  @param    nr     -  Nummer/indeks til en bank
 *  @return   Om 'kunde' har lån (true) i bank nr 'nr' eller ei (false)
 */
bool kundeLaanIBank(const struct Kunde* kunde, const int nr) { /* LAG INNMATEN */ }


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

/**
 *  Oppgave 2F - Alle EN kundes data skrives ut på fil.
 *  @param   ut     -  Filen det skal skrives til
 *  @param   kunde  -  Kunden som skrives til fil
 */
void kundeSkrivTilFil(FILE* ut, const struct Kunde* kunde) {   /* LAG INNMATEN */ }


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

/**
 *  Oppgave 2G - Leser ALT om EN kunde 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  */  }
```