Høgskolen i Gjøvik
Avdeling for teknologi

E K S A M E N

FAGNAVN:	**Grunnleggende datakunnskap og programmering**
FAGKODE:	**L 182 A**
EKSAMENSDATO:	**19. desember 2002**
KLASSE(R):	**02HIND*, 02HINE*, 02HDMU*, 02HING*, 02HGEOMAA, 02HSIV5**
TID:	**09.00-13.00**
FAGLÆRER:	**Frode Haug**

ANTALL SIDER UTLEVERT: 7 (inkludert denne forside)

TILLATTE HJELPEMIDLER: Kun læreboka ”OOP in C++”

- Kontroller at alle oppgavearkene er tilstede.

- Innføring med penn, eventuelt trykkblyant som gir gjennomslag. Pass på så du ikke skriver på mer enn
  ett innføringsark om gangen (da det blir uleselige gjennomslag når flere ark ligger oppå hverandre).

- Ved innlevering skilles hvit og gul besvarelse, som legges i hvert sitt omslag.

- Oppgavetekst, kladd og blå kopi beholder kandidaten til klagefristen er over.

- Ikke skriv noe av din besvarelse på oppgavearkene. Men, i oppgavetekst der du skal fylle ut svar i tegning/tabell/kurve, skal selvsagt dette innleveres sammen med hvit besvarelse.

- Husk kandidatnummer på alle ark. Ta vare på dette nummeret til sensuren faller.

Eksamenssettet består av tre ulike oppgavetyper:

Oppgave 1 omhandler teori fra datateknikk-delen.
Oppgave 2 omhandler hva som blir utskriften fra noen ulike program.
Oppgave 3 omhandler et litt større programmerings-case.

NB: De tre oppgavene er totalt uavhengige og kan derfor løses separat.

Oppgave 1 (30 %)

a) Internet:

a1) Forklar kort (max. ½ A4-ark) om synkrone/asynkrone diskusjoner og nevn noen eksempler på disse.

a2) Forklar kort (max. ½ A4-ark) om diskusjonsgrupper.

a3) Forklar kort (max. ½ A4-ark) om navnetjeneste.

b) Datamaskinen:

b1) Tegn datamaskinens overordnede arkitektur.

b2) Forklar kort (noen få setninger om hver) om:

b2a) Instruksjonspekeren (IP)

b2b) Primærlageret (RAM)

b2c) Kompilatorer

c) Datasikkerhet:

c1) Angi fem områder/typer for sårbarhet.

c2) Hvilke seks hovedområder har vi for "håndtering av virus-problematikken" ?

d) Personvern / etikk:

d1) Angi "de seks hensyn" man har ifm. personvern.

d2) Forklar kort (noen få setninger) om hva "relativistisk etikk" er.

Oppgave 2 (20 %)

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

```
#include <iostream>

using namespace std;

char txt[] = "LOMPEPAKKESAMLEMANI";

int main()  {
  bool fortsett = true;
  int i = 1, j;

  do  {
    cout << txt[++i * 2] << '\n';
  } while (i < 3);

  i = 3;
  while (fortsett)  {
    j = (i % 2) ? 3 : 5;
    i += j;
    cout << i << ": " << txt[i] << '\n';
    if (i > 13) fortsett = false;
  }
  return 0;
}
```

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

```
#include <iostream>

using namespace std;

void funk(char t[], int nn=2) {
  cout << t[nn/2] << '\n';
}

void funk(int & nn, char t[]) {
  nn /= 3;
  cout << t[nn] << '\n';
}

char funk(char t[], int mm, int nn) {
  return t[mm*nn];
}

int main()  {
  int n = 10, mm = 3;
  funk("BRILLEETUI", 10);
  funk(n, "BRILLEETUI");
  funk("BRILLEETUI");
  cout << funk("BRILLEETUI", mm, n) << '\n';
  cout << int(funk("BRILLEETUI", 2, 4)) << '\n';
  return 0;
}
```

Oppgave 3 (50 %)

NB: Les *hele* teksten for denne oppgaven *nøye*, før du begynner å besvare noe som helst. Studer vedlegget, som inneholder mange viktige opplysninger som du trenger/skal bruke. Legg spesielt merke til (og bruk) const'ene, struct'ene, globale variable og funksjonen "les_og_finn_kunde".

I denne oppgaven skal du lage et lite program som holder orden på bankkunder og deres kontoer.

Datastrukturen

For hver kunde (max.100 stk.) skal det lagres navn, adresse (gatenavn/-nr og postnr/-sted), antall kontoer vedkommende har (max.10 stk.) og disse kontoenes nummer og inneståend beløp.
På den første siden av vedlegget kan du se deklarasjoner/definisjoner av const'er, to struct'er og globale variable. *Dette skal være alt det du trenger av globale variable for å løse denne eksamensoppgaven!*
Datastrukturen består altså av arrayen "kunder" (med kunde-struct'er). Hver slik struct inneholder en array ("kontoer") med konto-struct'er. I alle arrayene er indeksene fom. nr.1 tom. nr."siste_kunde"/ "ant_kontoer" i bruk.

Oppgaven

a) **Lag funksjonen** void ny_kunde()

Funksjonen sjekker først om det er plass til flere kunder. Om så ikke er tilfelle, kommer det en melding om det. I motsatt fall så tas en ny indeks i bruk av arrayen "kunder" (ved å telle opp "siste_kunde"). Deretter leses vedkommendes navn og adresse inn. Nullstill også "ant_kontoer" hos vedkommende.

b) **Lag funksjonen** void ny_konto()

Funksjonen leser først inn et kundenavn og prøver å finne vedkommendes indeks i "kunder"-arrayen. Bruk funksjonen "les_og_finn_kunde" til å gjøre dette. Om vedkommende *ikke* er å finne, så skjer det ikke mer i funksjonen. I motsatt fall så sjekkes det om kunden har plass til flere kontoer. Om så *ikke* er tilfelle, kommer det en melding om dette. Eller så tas neste indeks i bruk av vedkommendes "kontoer"-array (ved å telle opp "ant_kontoer"), og kontonummeret og innskuddsbeløpet (som *må* være større eller likt null) leses inn for den nye kontoen.

c) **Lag funksjonen** void oversikt()

Funksjonen gjør innledningsvis det samme som "ny_konto" (prøver å finne kunden vha. "les_og_finn_kunde"). Om vedkommende er å finne så skrives adresse og antall kontoer på hver sin linje. Deretter skrives en linje for hver konto; inneholdende kontonummer og inneståend beløp/saldo.

d) **Lag funksjonen** void transaksjon()

Funksjonen gjør innledningsvis det samme som ”ny_konto” (prøver å finne kunden vha. ”les_og_finn_kunde”). Om vedkommende er å finne så sjekkes det om vedkommende har noen kontoer i det hele tatt. Om så ikke er tilfelle, kommer det en melding om dette. I motsatt fall så skrives alle numrene (*ikke* innestående beløp/saldo, dette forutsettes kjent av brukeren for eksempel vha. kommandoen ’O’) for bankkontoene og deres indeks i ”kontoer”-arrayen til skjermen. Brukeren spørres så om hvilken *indeks* vedkommende ønsker å endre på (**NB**: brukeren skal altså *ikke* skrive inn kontonummeret!). Om dette indekstallet er utenfor lovlig intervall, så kommer det en melding om dette. I motsatt fall så leses det inn et positivt (innskudd) eller negativt (uttak) beløp. Husk/pass på at saldo aldri får bli negativ! Til slutt blir det godtatte beløpet lagt til/trukket fra på aktuell konto.

NB: I de tre **oppgavene b) – d)** bør du også aller først ha en sjekk på om det i det hele tatt finnes noen kunder i datastrukturen. Om så ikke er tilfelle, gis meldingen ”Kartoteket er tomt!”, og resten av innmaten utføres ikke.

e) **Lag funksjonen** void les_fra_fil()

Funksjonen leser inn hele datastrukturen fra filen ”KUNTO.DTA”. Hver kunde er representert med en post på følgende format: Først en linje med vedkommendes navn, deretter en linje med adresse og til slutt en linje inneholdende ”ant_kontoer” og deretter parvis så mange kontoers nummer og beløp/saldo (*alle* disse tallene er altså på samme linje).

Klargjøring og forutsetninger

- Legg merke til at ”nr” i ”konto” er en int (vi antar at dets størrelse/lengde holder).
- Innestående beløp/saldo på en konto får aldri lov til å være negativ, dvs. her gis det ikke kreditt!
- I denne eksamensoppgaven skal det altså *ikke* lages funksjoner for å for eksempel endre/fjerne kunder og kontoer. Ei heller skal det lages funksjoner for å få oversikt over alle kundene (deres navn) eller å skrive dataene til fil.
- Gjør dine egne forutsetninger dersom du finner oppgaveteksten upresis eller ufullstendig. Gjør i så fall rede for disse forutsetningene *først* i besvarelsen din.

Lykke til !
frode@haug.com

Vedlegg: Halvferdig programkode (.tpl-fil)

```
                        // INCLUDE:
#include <iostream>             // cin, cout
#include <fstream>              // ifstream
#include <cstring>              // strcmp, strcpy
#include <cctype>               // toupper
using namespace std;
                        // CONST:
const int MAXKUNDER = 100;      // Max. antall kunder i datastrukturen.
const int MAXKONTO   =  10;     // Max. antall kontoer for EN kunde.
const int STRLEN     =  40;     // Max. lengde på tekststrenger.

                        // STRUCT:
struct konto  {                 // Konto:
  int nr;                       // Kontonummer.
  float belop;                  // Innestående beløp (saldo) på kontoen.
};

struct kunde  {                 // Kunde:
   char navn[STRLEN];           // Dens navn.
   char adr[STRLEN*2];          // Dens adresse (gate, postnr, poststed).
   int ant_kontoer;             // Antall aktive kontoer.
   konto kontoer[MAXKONTO+1];   // Kundens ulike kontoer.
};
                        // DEKLARASJON AV FUNKSJONER:
void  skriv_meny();             char  les_kommando();           int   les_og_finn_kunde();
void  ny_kunde();               void  ny_konto();               void  oversikt();
void  transaksjon();            void  les_fra_fil();

                        // GLOBALE VARIABLE:
int siste_kunde;
kunde kunder[MAXKUNDER+1];

int main()  {           // HOVEDPROGRAM:
   char kommando;
   les_fra_fil();               // Oppgave 3e
   skriv_meny();
   kommando = les_kommando();
   while (kommando != 'Q')  {
     switch (kommando)  {
       case 'N':  ny_kunde();     break; // Oppgave 3a
       case 'K':  ny_konto();     break; // Oppgave 3b
       case 'O':  oversikt();     break; // Oppgave 3c
       case 'T':  transaksjon();  break; // Oppgave 3d
       default:   skriv_meny();   break;
     }
     kommando = les_kommando();
   }
//   skriv_til_fil();
  return 0;
}
```

```
                        //  DEFINISJON AV FUNKSJONER:
void skriv_meny()  {                    //  Presenterer lovlige menyvalg:
  cout << "\n\nFØLGENDE KOMMANDOER ER LOVLIG:\n";
  cout << "\tN = registrer ny kunde\n";
  cout << "\tK = registrer ny konto hos en kunde\n";
  cout << "\tO = oversikt over en kundes kontoer\n";
  cout << "\tT = penger ut fra/inn på en konto\n";
  cout << "\tQ = quit/avslutt\n";
}


char les_kommando()  {                  //  Henter et ikke-blankt upcaset tegn:
  char ch;
  cout << "\n\nOppgi ›nske:  ";
  cin >> ch;   cin.ignore();
  return toupper(ch);
}


int les_og_finn_kunde()  {              //  Leser kundenavn og finner (om mulig):
  char nvn[STRLEN];                                                 // Kundens navn.
  cout << "\n\tKundens navn: ";  cin.getline(nvn, STRLEN);         // Leser navn.
  for (int i = 1;  i <= siste_kunde;  i++)                          // Går gjennom alle:
          if (strcmp(kunder[i].navn, nvn) == 0)  return i;          // Treff: returnerer.
  cout << "\n\tIngen kunde med dette navnet!\n";                   // Ingen treff.
  return 0;
}


void  ny_kunde()  {                     //  Legger inn (om plass) en ny kunde:
        //  Oppgave 3a:  Lag innmaten
}

void  ny_konto()  {                     //  Legger inn ny konto hos en eksisterende kunde:
        //  Oppgave 3b:  Lag innmaten
}

void  oversikt()  {                     //  Får oversikt over alt om en gitt kunde:
        //  Oppgave 3c:  Lag innmaten
}

void  transaksjon()  {                  //  Penger settes inn på/tas ut av en konto:
        //  Oppgave 3d:  Lag innmaten
}

void  les_fra_fil()  {                  //  Leser hele datastrukturen fra fil:
        //  Oppgave 3e:  Lag innmaten
}
```