NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamensoppgave i PROG1003
– Objekt-orientert programmering

Faglig kontakt under eksamen:	**Frode Haug**
Tlf:	**950 55 636**
Eksamensdato:	**16.mai 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 <iostream>
#include <string>
using namespace std;


class A {
  private:
    string txt;
    char ch;

  public:
    A(const string t, const char c) { txt = t;  ch = c; }
    virtual void display() const { cout << txt << ' ' << ch << ' '; }
    bool erLik(const string t) const { return (txt == t); }
};


class B : public A {
  private:
    string txt1, txt2;

  public:
    B(const string t1, const string t2, const string t3, const char c)
      : A(t3, c)  {  txt1 = t2;   txt2 = t1;  }
    virtual void display() const
      {  A::display();  cout << txt2 << ' ' << txt1;  }
    bool erLik(const string t) const
      {  return (txt1 == t  ||  txt2 == t);  }
    bool erLik(const B & b) const
      {  return (txt1 == b.txt1  ||  txt2 == b.txt2);  }
};


int main() {
    B* bobj1 = new B("Tore", "Brug", "Avik", 'x');
    bobj1->display();                                cout << '\n';

    A* bobj2 = new B("Anne", "Brud", "Haug", '-');
    bobj2->display();                                cout << '\n';

    cout << ((bobj1->erLik("Tore"))  ? "A&T" : "T&A")   << '\n';

    cout << (!(bobj2->erLik("Haug")) ? "A&T" : "T&A")   << '\n';

    cout << ((bobj1->erLik(*(dynamic_cast <B*> (bobj2)))) ? "A&T" : "T&A")
         << '\n';

    delete bobj1;   delete bobj2;

    return 0;
}
```

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

```
#include <iostream>
#include <vector>
#include <list>
using namespace std;

int main() {
    vector <char> tegn1(4);
    vector <char> tegn2 { 'A', 'N', 'E', 'T', 'O', 'R', 'E' };
    vector <char> tegn3(8, '-');
    vector <char> tegn4(tegn2);

    tegn1 = tegn3;
    for (const auto & val : tegn1) cout << ' ' << val;   cout << " : ";
    for (const auto & val : tegn4) cout << ' ' << val;   cout << '\n';

    auto it = tegn2.begin();
    for (  ;  *it != 'O';  it++)
        cout << ' ' << *it;
    cout << ' ' << tegn2[3] << ' ' << tegn2[tegn2.size()-1]   << '\n';

    tegn2.erase(it, tegn2.end());
    tegn2.pop_back();       tegn2.pop_back();
    tegn2.push_back('N');   tegn2.push_back('E');
    for (const auto & val : tegn2) cout << ' ' << val;   cout << '\n';


    list <int> tall(3, 7);
    tall.push_front(2);   tall.push_front(4);
    tall.push_back(1);    tall.push_back(3);
    for (const auto & val : tall) cout << ' ' << val;
    cout << " - " << tall.front() << ' ' << tall.back()       << '\n';

    tall.reverse();       tall.remove(7);
    for (auto it = tall.rbegin();  it != tall.rend();  it++)
        cout << ' ' << *it;                                cout << '\n';

    return 0;
}
```

Oppgave 2 (70%)

Les *hele* teksten for denne oppgaven (2a-2j, dvs. 10 oppgaver) *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 `consten`, `enum`en m/kommentar, klassene med *og uten* datamedlemmer og (*ferdiglagde*) funksjoner (inni/utenfor klassene, og at de er både `private`, `protected` og `public`), globale variable, `main` **og** `skrivMeny()`**.**
Husk også på de ferdiglagde funksjonene på `LesData2.h`**. Bruk *alt* dette *svært* aktivt.**

Vi skal lage et program som holder orden på ulike personer (barn og voksne) sine uttak/kjøp av varer på ett eller annet sted (arrangement/fornøyelsespark/dyrepark/lekeland). Dette fungerer slik at personer (alle med et *unikt* nummer/ID) går rundt på det aktuelle stedet og bare henter/tar ut ulike varer fra ulike boder/kiosker/bord. De viser sitt nummer (som f.eks. står på et armbånd, plastkort eller app), og så blir varen/uttaket/kjøpet registrert i vedkommendes `kjopene`. Når personen drar/forlater stedet, gjøres regningen opp (personen betaler for alle sine kjøp/uttak).

Datastrukturen

Datastrukturen består kun av *listen* `gPersonene`. Denne inneholder i praksis enten `Barn`- eller `Voksen`-objekter (begge subklasser av `Person`). Alle personene har et *unikt* nummer/ID, som `gSisteNummer` automatisk tildeler dem, ved at den telles opp hver gang en ny person lages.

Oppgaven

a) **Skriv innmaten til** `void Person::lesData()`
Leser inn dataene til personens *to siste private* datamedlemmer.

b) **Skriv innmaten til** `void nyPerson()`
Leser først inn (og sikrer) om det er et barn eller en voksen som skal legges inn. Personens nye unike nummer telles opp og skrives ut på skjermen. Aktuell person opprettes (og får sitt nummer), vedkommendes to datamedlemmer leses inn, og personen legges inn i datastrukturen.
NB: Resortering av lista er overflødig, da nye med høyere numre *alltid* legges inn bakerst i lista.

c) **Skriv innmaten til de to virtuelle** `void lesKjop()` **i både** `Barn` **og** `Voksen`
Det leses og sikres et *lovlig* kjøp/uttak for vedkommende person. Dette registreres så i vedkommendes `kjopene` (bruk ferdiglaget funksjon). Lovlige kjøp/uttak for
`Barn` er: **P**ølse, Pop**C**orn, **G**odtepose, **S**jokolade, **I**s og **B**rus
`Voksen` er: **H**amburger, **K**affe, **S**jokolade, **I**s og **B**rus
Barn og voksne får altså *ikke* lov til å kjøpe/hente/ta ut *eksakt de samme* varene.

d) **Skriv innmaten til** `void personKjoperEnVare()`
Det kommer en egen melding om det er tomt for personer. I motsatt fall leses et *relevant* nummer (mellom 1 og `gSisteNummer`). Om nummeret *ikke* er å finne lengre, kommer en egen melding. I motsatt fall registreres det er kjøp/uttak hos vedkommende.

e) **Skriv innmaten til** `int Person::antallKjop(…)`
Returner antall kjøp/uttak hos personen av varen angitt som parameter.

f) **Skriv innmaten til** `void Person::skrivData()` **og**
`void Person::skrivKjop(…)`

Skriver ut alle personens tre datamedlemmer på skjermen. Dersom personen foreløpig ikke har foretatt noen kjøp/uttak, så skrives en egen melding. I motsatt fall skrives (vha. flere kall til den andre funksjonen og den i 2e) antall kjøp/uttak (når ulikt 0 (null)) av hver av de ulike varene.
NB: Her trengs det ikke å ta hensyn til om det er barn eller voksen. For ønsker man f.eks. å skrive antall pølse-kjøp av en voksen, så kommer det ikke noen utskrift – for dette er alltid 0.

g) **Skriv innmaten til de to virtuelle** `int skalBetale()` **i både** `Barn` **og** `Voksen`
Funksjonen returnerer *totalsummen* personen skal betale for alle kjøpene. Gitt at alle varer koster `PRIS`, unntatt Hamburger som koster et dobbelte. **Hint:** Bruk funksjonen i 2e.

h) **Skriv innmaten til** `int avreisePerson()`
Egen melding om det er *tomt* for personer. Ellers leses det et *relevant* nummer. Finnes *ikke* denne kommer det også en melding. I motsatt fall skrives alle vedkommendes data *og hva personen totalt skal betale* ut på skjermen. Til slutt slettes/fjernes personen fra datastrukturen (husk å unngå memory-lekkasje). **NB:** Husk at funksjonen skal returnere hva vedkommende total skal betale (dette brukes/utnyttes av funksjonen i 2i).

i) **Skriv innmaten til** `void avreiseFamilie()`
Leser og *oppsummerer* hva ulike personer skal betale inntil vedkommendes sum er 0 (dvs. at det er tomt for personer, at vedkommende nummer ikke er å finne, eller personen har null kjøp/uttak). Til slutt skrives totalsummen for hva gruppen/familien skal betale ut.

j) **Skriv innmaten til** `void lesFraFil()` **og** `Person::Person(…… inn)`
Disse funksjonene sørger til sammen for at *hele* listen med personer leses inn fra filen «KJOP.DTA». Aller først på filen ligger `gSisteNummer` og antall personer som videre kommer på filen (disse to tallene er gjerne ulike, da personer har avreist/dratt). Formatet for resten av filen bestemmer du helt selv, men **dette *skal* oppgis som en del av besvarelsen**.
NB1: Listen skal være sortert etterpå (for vi har ingen garanti for at den er det på filen).
NB2: Se hvordan `Person::Person(… inn)` kalles via subklassenes constructor.
Hint: Bruk gjerne den ferdiglagde funksjonen `registrerEttKjop(…)`

Annet (klargjørende):

- Det er *ikke* en del av denne eksamensoppgaven å lage kode som *skriver hele listen til fil*.
- Du *skal* bruke `LesData2.h` ifm. løsningen av denne oppgaven. Du får nok også bruk for (deler av) pensumets temaer innen STL, men *ikke* bruk saker fra STL, templates eller stoff/biblioteker utenfor pensum.
- Gjør dine egne forutsetninger og presiseringer av oppgaven, dersom du skulle finne dette nødvendig. Gjør i så fall klart rede for disse der det gjelder i besvarelsen din av oppgaven(e).

Lykke til med *uttak* av en god karakter!

FrodeH

Vedlegg til PROG1003, 16.mai 2022: Halvferdig programkode

```
#include <iostream>                 //  cout, cin
#include <fstream>                  //  ifstream, ofstream
#include <iomanip>                  //  setw
#include <string>
#include <vector>
#include <list>
#include <algorithm>                //  Du får sikkert bruk for noen av disse.
#include "LesData2.h"               //  Verktøykasse for lesing av diverse data
using namespace std;


const int PRIS =  20;               ///<  Den vanligste prisen pr.enhet/ting.


/**
 *  Vare (de ulike varene brukerne totalt kan hente ut/kjøpe).
 */
enum Vare { polse, popcorn, godtepose,    //  Menyvalg:   P  C  G
            sjokolade, is, brus,          //              S  I  B
            hamburger, kaffe };           //              H  K


/**
 *  Person (med UNIKT nummer, navn, mobilnr og ALLE vedkommendes kjøp).
 */
class Person {
  private:
    int nummer,                                 //  ID - sortert på dette.
        mobil;
    string navn;

    void skrivKjop(const Vare vare) const;      //  Oppgave 2F

  protected:
    vector <Vare> kjopene;                      //  ALLE kjøpene/uthentinger.

    int  antallKjop(const Vare vare) const;     //  Oppgave 2E
    void registrerEttKjop(const char kjop);     //        (Ferdiglaget)

  public:
    Person(const int nr)  { nummer = nr;   }    //        (Ferdiglaget)
    Person(ifstream & inn);                     //  Oppgave 2J
    int  hentID() const   { return nummer; }    //        (Ferdiglaget)
    void lesData();                             //  Oppgave 2A
    void skrivData() const;                     //  Oppgave 2F
    virtual void lesKjop() = 0;                 //  Oppgave 2C - Pure virtual,
                                                //     dvs. "barna" MÅ lage den.
    virtual int  skalBetale() const = 0;        //  Oppgave 2G - Pure virtual.
};


/**
 *  Barn (UTEN egne datamedlemmer, men med virtuelle funksjoner).
 */
class Barn : public Person {
  public:
    Barn(const int nr)   : Person(nr)  {  }     //        (Ferdiglaget)
    Barn(ifstream & inn) : Person(inn) {  }     //        (Ferdiglaget)
    virtual void lesKjop();                     //  Oppgave 2C
    virtual int  skalBetale() const;            //  Oppgave 2G
};
```

```
/**
 *  Voksen (UTEN egne datamedlemmer, men med virtuelle funksjoner).
 */
class Voksen : public Person {
  public:
    Voksen(const int nr)   : Person(nr)  {  } //        (Ferdiglaget)
    Voksen(ifstream & inn) : Person(inn) {  } //        (Ferdiglaget)
    virtual void lesKjop();                   //  Oppgave 2C
    virtual int  skalBetale() const;          //  Oppgave 2G
};


void avreiseFamilie();                        //  Oppgave 2I
int  avreisePerson();                         //  Oppgave 2H
void lesFraFil();                             //  Oppgave 2J
void nyPerson();                              //  Oppgave 2B
void personKjoperEnVare();                    //  Oppgave 2D
void skrivMeny();


int  gSisteNummer = 0;                   ///<  Siste FORTLØPENDE nummer brukt.
list <Person*> gPersonene;               ///<  ALLE personene som kjøper noe.


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

  lesFraFil();                                      //  Oppgave 2J

  skrivMeny();
  valg = lesChar("\nKommando");

  while (valg != 'Q') {
     switch (valg) {
       case 'N': nyPerson();              break;  //  Oppgave 2B
       case 'K': personKjoperEnVare();    break;  //  Oppgave 2D
       case 'E': avreisePerson();         break;  //  Oppgave 2H
       case 'F': avreiseFamilie();        break;  //  Oppgave 2I
       default:  skrivMeny();             break;
     }
     valg = lesChar("\nKommando");
  }

  cout << "\n\n";
  return 0;
}



// ------------------------------------------------------------------------
//                        DEFINISJON AV KLASSE-FUNKSJONER:
// ------------------------------------------------------------------------

/**
 *  Oppgave 2J - Leser inn ALLE egne data fra fil.
 *
 *  @param  inn   -  Filobjektet det leses inn data fra
 */
Person::Person(ifstream & inn) {                    /*   LAG INNMATEN   */  }
```

```
/**
 *  Oppgave 2E - Finner og returnerer antall kjøp/uthentinger av en gitt vare.
 *
 *  @param    vare   -  Varen det skal sjekkes antall kjøp/uthentinger av
 *  @return   Antall kjøp/uthentinger av den gitte varen
 */
int Person::antallKjop(const Vare vare) const  {     /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2A - Leser inn ALLE egne data fra tastaturet.
 */
void Person::lesData()  {                            /*   LAG INNMATEN   */  }


/**
 *  Legger inn medsendt kjøp (enum) i egen datastruktur/vector.
 *
 *  @param    kjop   -  kjøpet/uthentingen som skal unnalagres
 */
void Person::registrerEttKjop(const char kjop)  {
  switch (kjop)  {
    case 'P':  kjopene.push_back(polse);      break;
    case 'C':  kjopene.push_back(popcorn);    break;
    case 'G':  kjopene.push_back(godtepose);  break;
    case 'S':  kjopene.push_back(sjokolade);  break;
    case 'I':  kjopene.push_back(is);         break;
    case 'B':  kjopene.push_back(brus);       break;
    case 'H':  kjopene.push_back(hamburger);  break;
    case 'K':  kjopene.push_back(kaffe);      break;
    default:   cout << "\n\tUlovlig vare-type som parameter!\n\n";  break;
  }
}


/**
 *  Oppgave 2F - Skriver ALLE egne data ut på skjermen.
 */
void Person::skrivData() const  {                    /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2F - Skriver ut på skjermen ANTALL kjøp/uttak av en gitt vare.
 *
 *  @param    vare   -  Varen det skal skrives ut antallet av
 */
void Person::skrivKjop(const Vare vare) const  {     /*   LAG INNMATEN   */  }

// ------------------------------------------------------------------------

/**
 *  Oppgave 2C - Registrerer ETT kjøp/uthenting av barnet.
 */
void Barn::lesKjop() {                               /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2G - Beregner hvor mye som skal betales for alle kjøpene/uttakene.
 *
 *  @return   Totalt antall kroner som alle kjøpene/uttakene koster
 */
int Barn::skalBetale() const  {                      /*   LAG INNMATEN   */  }
```

```
// ---------------------------------------------------------------------------

/**
 *  Oppgave 2C - Registrerer ETT kjøp/uthenting av den voksne.
 */
void Voksen::lesKjop() {                                /*   LAG INNMATEN   */ }


/**
 *  Oppgave 2G - Beregner hvor mye som skal betales for alle kjøpene/uttakene.
 *
 *  @return   Totalt antall kroner som alle kjøpene/uttakene koster
 */
int Voksen::skalBetale() const  {                       /*   LAG INNMATEN   */ }

// ---------------------------------------------------------------------------
//                       DEFINISJON AV ANDRE FUNKSJONER:
// ---------------------------------------------------------------------------

/**
 *  Oppgave 2I - En hel familie/gruppe sjekker ut/reiser.
 */
void avreiseFamilie()  {                                /*   LAG INNMATEN   */ }


/**
 *  Oppgave 2H - EN enkelt person sjekker ut/reiser, finner hva skal betales.
 *
 *  @return   Summen vedkommende skal betale for sine kjøp/uttak
 */
int avreisePerson()  {                                  /*   LAG INNMATEN   */ }


/**
 *  Oppgave 2J - Leser ALLE personene (barn/voksne) inn fra fil.
 */
void lesFraFil()  {                                     /*   LAG INNMATEN   */ }


/**
 *  Oppgave 2B - Legger inn en ny person (barn eller voksen).
 */
void nyPerson() {                                       /*   LAG INNMATEN   */ }


/**
 *  Oppgave 2D - Person kjøper/henter ut EN vare.
 */
void personKjoperEnVare()  {                            /*   LAG INNMATEN   */ }


/**
 *  Skriver programmets menyvalg/muligheter på skjermen.
 */
void skrivMeny()  {
  cout << "\nFølgende kommandoer er tilgjengelige:\n"
       << "   N  - Ny person (barn eller voksen)\n"
       << "   K  - en person Kjøper/henter en vare\n"
       << "   E  - avreise/utsjekk av En person\n"
       << "   F  - avreise/utsjekk av en hel Familie/gruppe\n"
       << "   Q  - Quit / avslutt\n";
}
```