NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Kontinuasjonseksamensoppgave i PROG1003 – Objekt-orientert programmering

Faglig kontakt under eksamen:	**Frode Haug**
Tlf:	**950 55 636**
Eksamensdato:	**15.august 2024**
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):	**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>
#include <algorithm>
using namespace std;

class A  {
  protected:
    string aa;
  public:
    A(string s)  {  aa = s;  }
    virtual void display() = 0;
};

class B : public A  {
  private:
    int bb;
  public:
    B(string s, int b) : A(s)  {  bb = b;  }
    void display()  {  cout << aa << ' ' << bb;  }
    void funk1(string s)  {  aa += (' ' + s);  }
    int  funk2(string s, string t)
              {  aa.insert(aa.size(),s);  return (aa.find_first_of(t));  }
};

class C : public A  {
  private:
    char cc;
  public:
    C(string s, char c) : A(s)  {  cc = c;  }
    void display()  {  cout << aa << ' ' << cc;   }
    string funk1()  {  return (aa.substr(3, 3));  }
    void funk2(string s)  { for (int i = 73; i < 77;  i++) aa += s;   }
    bool funk2(char c)  { return (count(aa.begin(), aa.end(), c) > 4); }
};

int main() {
  B  bObj("Brokke", 1708);
  C* cObj = new C("Hovden", 'T');
  A* obj = new C("Revesand", 'E');
  bObj.display();  cout << ' ';  cObj->display(); cout << ' ';
  obj->display();  cout << '\n';
  bObj.funk1("hytte");  bObj.display();  cout << '\n';
  cout << cObj->funk1() + "ne" << '\n';
  cout << bObj.funk2("felt", "t") << '\n';
  cObj->funk2("AT"); cObj->display(); cout << ' '<< cObj->funk2('T') << '\n';

  return 0;
}
```

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

```
#include <iostream>
#include <string>
#include <vector>
#include <map>
using namespace std;


void skriv(map <int, string> & m)  {
  for (const auto & val : m)
      cout << val.first << ':' << val.second << "  ";    cout << '\n';
}


int main() {
  vector <string> txt { "AA", "LL", "NN", "SS", "RR", "EE", "AA" };
  map <int, string> mapen;

  for (int i = txt.size()-1;  i >= 0;   i--)
      mapen[i*3] = txt[i];

  skriv(mapen);

  auto it = mapen.begin();
  mapen.erase(++it);
  skriv(mapen);

  auto it2 = mapen.rbegin();
  for (int i = 0;  i < 4;  i++)  it2++;
  while (it2 != mapen.rend()) cout << (it2++)->first << ' ';   cout << '\n';


  it  = mapen.lower_bound(6);
  auto it3 = mapen.upper_bound(15);

  cout << it3->first << '\n';

  for (  ;  it != it3;  )
      cout << (it++)->first << ' ';   cout << '\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 `const`'ene, klassene med datamedlemmer og (*ferdiglagde*) funksjoner inni klassen, global variabel, `main`, `skrivMeny` og `finnEnGittBil`. Legg spesielt merke til hvilke funksjoner som er virtuelle eller ei inni klassene. Husk også på funksjonene på `LesData2.h`. Bruk *alt* dette *svært* aktivt.

Det skal holdes orden på ulike biler/bobiler og hvilke byer de har vært i et visst antall ganger.

Datastrukturen

Datastrukturen består *kun* av *listen* `gBilene`. Alle bilene har alle et *unikt* registreringsnummer (regnr), og skal være sortert på dette. I listen ligger det objekter av begge typene `Bil` og `Bobil`. Inni hver ligger det bl.a. to vectorer med henholdsvis navnet på byene som (bo)bilen har vært i, og antall ganger bilen ha vært i vedkommende by. Indeks nr. `i` i begge arrayene er sammenhørende verdier.

Oppgaven

a)

Skriv innmaten til `void skrivAlleBiler()` **og** `void Bil::skrivData()`

Det kommer en egen melding om ingen biler finnes. Ellers skrives antall biler ut på skjermen, samt regnr, merke og *kun* antall ulike byer som er besøkt av hver av bilene. En linje pr.bil.

b)

Skriv innmaten til
```
void skrivEnGittBil()
virtual void Bil::skrivData2()
virtual void Bobil::skrivData2()
```

Den første funksjonen leser først inn et ønsket regnr. Deretter blir det lett etter denne bilen (bruk ferdiglaget funksjon). Finnes den *ikke*, kommer det en egen melding. I motsatt fall skrives *alt* om vedkommende bil, ut fra hvilken type den er av (vha. de to andre virtuelle funksjonene), også en oversikt over byene den har besøkt og antall ganger den har vært i vedkommende by.
NB: Den aller siste funksjonen bør nok bl.a. også kalle den andre funksjonen.

c)

Skriv innmaten til
```
void nyBil()
virtual void Bil::lesData()
virtual void Bobil::lesData()
```

Den første funksjonen spør først om den nye bilens regnr. Finnes denne allerede, kommer det en melding. I motsatt fall leses *og sikres* det om brukeren ønsker å opprette en `Bil` ('I') eller en `Bobil` ('O'). Aktuelt bilobjekt opprettes, *alle* dens aktuelle data leses inn (*unntatt* byer besøkt, det skal gjøres i 2d) vha. de to virtuelle funksjonene, den legges inn bakerst i datastrukturen, og listen sorteres til slutt korrekt bl.a. vha. lamda-funksjon.

d)

Forklar `void byBesok()` **og skriv innmaten til** `void Bil::bybesok()`

Den første funksjonen gjør omtrent det samme som den første i oppgave 2b.
Forklar kort hva forskjellen er. Den andre funksjonen skriver først og til slutt ut *alle* bilens data. Mellom dette spør den om bynavn, inntil brukeren kun svarer ENTER (en tom tekst). For hvert bynavn blir det sjekket om den allerede finnes i vectoren for bilens byer. Gjør den det, så telles antall besøk i den opp med 1 (en). Ellers legges den inn som ny bakerst i vedkommende `vector`, og antall besøk i den registreres som 1 (en).

e) **Skriv innmaten til** `void skrivTotaltAntallBybesok()`
og all annen kode som trengs for å få skrevet ut *alle* byene (*sortert alfabetisk*) og *totalt* antall besøk i hver av dem, gjort av *alle* bilene til sammen. (Full score på denne oppgaven gis *kun* ved mye bruk av STL-biblioteket og ingen(!) if-setninger.)

f) **Skriv innmaten til**
```
void skrivTilFil()
void Bil::skrivTilFil2(…)
virtual void Bil::skrivTilFil(…)
virtual void Bobil::skrivTilFil(…)
```
Disse funksjonene sørger til sammen for at *hele* datastrukturen skrives ut til filen «BILER.DTA». De to funksjonene *uten* '2' i bør skrive det som er spesielt for *kun* vedkommende klasse, mens den med '2'-tallet skriver det som er felles for dem (dvs. *alt* under `private` i `Bil`). Formatet på filen bestemmer du helt selv, men **det *skal* oppgis som en del av besvarelsen**.

g) **Skriv innmaten til** `void lesFraFil()`
`Bil::Bil(ifstream …)` **og** `Bobil::Bobil(ifstream …)`
Disse funksjonene sørger til sammen for at *hele* datastrukturen leses inn fra filen gitt ovenfor, og med det formatet du selv bestemt. Du trenger *ikke* å sortere dem her, da vi forutsetter at de er skrevet ut sortert til fil.

Annet (klargjørende):

- Regnr og bynavn er *alltid ett* ord. `Merke` *kan* bestå av flere ord.
  Vectoren `bynavn` inni hver `Bil` er *ikke* sortert alfabetisk.
  Du trenger ikke å ha noen sjekk på et regnr sin gyldighet (at de i Norge *må* bestå av to bokstaver etterfulgt av fem sifre).
- Det er masse data som *ikke* registreres om hver enkelt bil, f.eks. om de har air-condition, manuelt/automatisk gir, elbil/hybrid/fossil eller plass til N kofferter.
- 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).

God (bo)biltur !

FrodeH

Vedlegg til PROG1003, august 2024: Halvferdig programkode

```
#include <iostream>                //  cout, cin
#include <fstream>                 //  ifstream, ofstream
#include <string>
#include <vector>
#include <list>
#include <algorithm>               //  find_if
#include "LesData2.h"              //  Verktøykasse for lesing av diverse data
using namespace std;

const int MINPERS  =    2;         ///<  Bil minimum antall personer.
const int MAXPERS  =    8;         ///<  Bil maksimum antall personer.
const int MINLEN   =  4.0;         ///<  Bobils minimumslengde.
const int MAXLEN   = 10.0;         ///<  Bobils maksimumslengde.

/**
 *  Bil med dets registreringsnummer, merke/type (f eks Audi Q4) og
 *    antall personer det er plass til i bo(bilen), samt vector med navnet
 *    på besøkte byer og vector med antall besøk i aktuell by.
 */
class Bil  {
  private:
    string regnr,
           merke;
    int antallPersoner;
    vector <string> bynavn;
    vector <int>    antallBesok;
  public:
    Bil(string rnr)  { regnr = rnr;  antallPersoner = 0; } //    (Ferdiglaget)
    Bil(ifstream & inn);                                   //  Oppgave 2G
    void bybesok();                                        //  Oppgave 2D
    string hentID()    { return  regnr;  }                 //    (Ferdiglaget)
    virtual void lesData();                                //  Oppgave 2C
            void skrivData()   const;                      //  Oppgave 2A
    virtual void skrivData2()  const;                      //  Oppgave 2B
    virtual void skrivTilFil(ofstream & ut) const;         //  Oppgave 2F
            void skrivTilFil2(ofstream & ut) const;        //  Oppgave 2F
};

/**
 *  Avledet klasse 'Bobil' med bobilens lengde og antall sengeplasser.
 */
class Bobil : public Bil  {
  private:
    float lengde;
    int   antallSengeplasser;
  public:                                                  //   (Ferdiglaget):
    Bobil(string rnr) : Bil(rnr)  {  lengde = 0.0;  antallSengeplasser = 0;  }
    Bobil(ifstream & inn);                                 //  Oppgave 2G
    virtual void lesData();                                //  Oppgave 2C
    virtual void skrivData2()  const;                      //  Oppgave 2B
    virtual void skrivTilFil(ofstream & ut) const;         //  Oppgave 2F
};


void byBesok();                                            //  Oppgave 2D
Bil* finnEnGittBil(string regnr);                          //    (Ferdiglaget)
void lesFraFil();                                          //  Oppgave 2G
void nyBil();                                              //  Oppgave 2C
void skrivAlleBiler();                                     //  Oppgave 2A
void skrivEnGittBil();                                     //  Oppgave 2B
void skrivMeny();                                          //    (Ferdiglaget)
void skrivTilFil();                                        //  Oppgave 2F
void skrivTotaltAntallBybesok();                           //  Oppgave 2E
```

```
list <Bil*> gBilene;                    ///<  Datastrukturen med ALLE bilene.


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

  lesFraFil();                                                  //  Oppgave 2G

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

  while (valg != 'Q') {
     switch (valg) {
       case 'A': skrivAlleBiler();             break;       //  Oppgave 2A
       case 'G': skrivEnGittBil();             break;       //  Oppgave 2B
       case 'N': nyBil();                      break;       //  Oppgave 2C
       case 'B': byBesok();                    break;       //  Oppgave 2D
       case 'T': skrivTotaltAntallBybesok();   break;       //  Oppgave 2E
       default:  skrivMeny();                  break;
     }
     valg = lesChar("\nKommando");
  }

  skrivTilFil();                                                //  Oppgave 2F

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


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

/**
 *  Oppgave 2G - Leser inn ALLE data om EN bil fra fil.
 *
 *  @param    inn  -  Filen det leses inn fra
 */
Bil::Bil(ifstream & inn)  {                     /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2D - Leser inn null eller flere (nye) besøkte byer av bilen..
 *
 *  @see   Bil::skrivData2()
 */
void Bil::bybesok()  {                          /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2C - Leser inn ALT om en bil (unntatt besøkte byer).
 */
void Bil::lesData()  {                          /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2A - Skriver ut på skjermen noen utvalgte HOVEDdata.
 */
void Bil::skrivData()   const  {                /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2B - Skriver ALT om en bil ut på skjermen.
 */
void Bil::skrivData2()  const  {                /*    LAG INNMATEN    */      }
```

```
/**
 *  Oppgave 2F - Skriver ut på fil ALLE bilens data.
 *
 *  @param    ut  -  Filen det skal skrives til
 *  @see      Bil::skrivTilFil(...)
 */
void Bil::skrivTilFil(ofstream & ut) const  {  /*    LAG INNMATEN    */     }

/**
 *  Oppgave 2F - Skriver ut på fil ALLE bilens data.
 *
 *  @param    ut  -  Filen det skal skrives til
 */
void Bil::skrivTilFil2(ofstream & ut) const  { /*    LAG INNMATEN    */     }
// ---------------------------------------------------------------------------
/**
 *  Oppgave 2G - Leser inn ALLE data om en bobil fra fil.
 *
 *  @param   inn  -  Filen det leses inn fra
 *  @see     Bil::Bil(...)
 */
Bobil::Bobil(ifstream & inn) : Bil(inn)  {     /*    LAG INNMATEN    */     }

/**
 *  Oppgave 2C - Leser inn ALT om en bobil.
 *
 *  @see     Bil::lesData()
 */
void Bobil::lesData()  {                       /*    LAG INNMATEN    */     }

/**
 *  Oppgave 2B - Skriver ALT om en bobil ut på skjermen.
 *
 *  @see     Bil::skrivData2()
 */
void Bobil::skrivData2()  const  {             /*    LAG INNMATEN    */     }

/**
 *  Oppgave 2F - Skriver ut på fil ALLE bobilens data.
 *
 *  @param    ut  -  Filen det skal skrives til
 *  @see      Bil::skrivTilFil(...)
 */
void Bobil::skrivTilFil(ofstream & ut) const  {/*    LAG INNMATEN    */     }
// ---------------------------------------------------------------------------
//                       DEFINISJON AV ANDRE FUNKSJONER:
// ---------------------------------------------------------------------------
/**
 *  Oppgave 2D - Legger inn nye bybesøk for en gitt bil.
 *
 *  @see   finnEnGittBil(...)
 *  @see   Bil::bybesok()
 */
void byBesok()  {                              /*    LAG INNMATEN    */     }

/**
 *  Returnerer (om mulig) en peker til bil med ønsket regnr.
 *
 *  @param    regnr  -  Regnr det skal letes/søkes etter
 *  @return   Peker til søkt bil eller 'nullptr'
 *  @see      Bil::hentID()
 */
Bil* finnEnGittBil(string regnr)  {              //  Leter etter en gitt bil:
  auto it = find_if(gBilene.begin(), gBilene.end(),[regnr] (const auto & val)
                                     { return (val->hentID()  == regnr);  } );
  return ((it != gBilene.end()) ? *it : nullptr); //  Returnerer peker til
}                                                 //    bilen eller 'nullptr'.
```

```
/**
 *  Oppgave 2G - Leser ALLE bilene inn fra fil.
 *
 *  @see   Bil::Bil(...)
 *  @see   Bobil::Bobil(...)
 */
void lesFraFil()  {                          /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2C - Legger (om mulig) inn en ny (bo)bil i datastrukturen.
 *
 *  @see   finnEnGittBil(...)
 *  @see   Bil::Bil(...)
 *  @see   Bobil::Bobil(...)
 */
void nyBil()  {                              /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2A - Skriver ut på skjermen hoveddatene om ALLE bilene.
 *
 *  @see   Bil::skrivData()
 */
void skrivAlleBiler()  {                     /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2B - Skriver ut på skjermen ALLE data om en GITT bil.
 *
 *  @see   finnEnGittBil(...)
 *  @see   Bil::skrivData2()
 *  @see   Bobil::skrivData2()
 */
void skrivEnGittBil()  {                     /*    LAG INNMATEN    */      }

/**
 *  Skriver programmets menyvalg/muligheter på skjermen.
 */
void skrivMeny()  {
  cout << "\nFolgende kommandoer er tilgjengelige:\n"
       << "   A  - skriv Alle biler\n"
       << "   G  - skriv alt om en Gitt bil\n"
       << "   N  - ny (bo)bil\n"
       << "   B  - (ny) By er besokt\n"
       << "   T  - skriver Totalt antall besok i hver by\n"
       << "   Q  - Quit / avslutt\n";
}

/**
 *  Oppgave 2F - Skriver ALLE bilene ut til fil.
 *
 *  @see   Bil::skrivTilFil(...)
 *  @see   Bobil::skrivTilFil(...)
 */
void skrivTilFil()  {                        /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2E - Skriver ALLE byene og TOTALT antall besøk i hver av dem.
 *
 *  @see   Bil::hentByer(...)
 */
void skrivTotaltAntallBybesok() {            /*    LAG INNMATEN    */      }
```