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:	**12.august 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 (28%)

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

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


class A {
  protected:
    int aa1, aa2;
  public:
    A()  { aa1 = 30;  aa2 = 17;  }
    A(const int a1, const int a2)  {  aa1 = a1;  aa2 = a2;  }
    virtual void display() const   {  cout << aa2 << ' ' << aa1 << ' ';  }
    virtual bool erLik(const int aa) const  {  return (aa1 == aa);  }
};


class B : public A {
  private:
    int bb1, bb2;
  public:
    B(const int b1, const int b2) { bb1 = b1;  bb2 = b2; }
    B(const int b1, const int b2, const int b3, const int b4) : A(b3, b2)
       {  bb1 = b4;  bb2 = b1;  }
    void display() const  {  A::display();  cout << bb1 << ' ' << bb2;  }
    bool erLik(const int bb) const
                        {  return(aa1 % 20 == bb  ||  bb1 % 10 == bb);  }
};


int main() {
    A ob1, ob2(17, 8);      ob1.display();  ob2.display();    cout << '\n';
    B* ob3 = new B(30, 5);           ob3->display();          cout << '\n';
    B* ob4 = new B(29, 6, 26, 9);    ob4->display();          cout << '\n';
    cout << (ob2.erLik(17)) << ' ' << (!ob1.erLik(17)) << '\n';
    cout << (ob4->erLik(6)) << ' ' << (ob3->erLik(4))  << '\n';
    return 0;
}
```

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

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


int main() {
    vector <int> tall1(2);
    vector <int> tall2 { 2, 4, 8, 16, 32, 64 };
    vector <int> tall3(5, 9);
    vector <int> tall4(tall2);

    tall1 = tall3;
    for (const auto & val : tall1) cout << ' ' << val;   cout << " : ";
    for (const auto & val : tall4) cout << ' ' << val;   cout << '\n';

    cout << tall2[3] << ' ' << tall2.front() << ' '
         << tall2.back() << ' ' << tall2[tall2.size()-2]      << '\n';

    auto it  = find(tall2.begin(),  tall2.end(),  16);
    auto it2 = find(tall4.begin(),  tall4.end(),  16);

    cout << "Verdier er ";
    if (it == tall2.end()  ||  it2 == tall4.end()  ||  *it != *it2)
        cout << "IKKE ";                              cout << "like.\n";


    map <string, string> mengde;
    mengde.insert(pair <string, string>("10A", "Luton"));
    mengde.insert(pair <string, string>("11A", "Tottenham"));
    mengde.insert(pair <string, string>("12A", "Chelsea"));
    mengde.insert(pair <string, string>("13A", "Wolves"));
    mengde["10B"] = "Barnsley";
    mengde["11B"] = "Everton";
    mengde["12B"] = "Watford";
    mengde["13B"] = "Arsenal";

    for (const auto & val : mengde)
        cout << ' ' << val.first;                        cout << '\n';

    for (auto it = mengde.find("12A");  it != mengde.end();  it++)
        cout << ' ' << it->second;                       cout << '\n';


    return 0;
}
```

Oppgave 2 (72%)

Les *hele* teksten for denne oppgaven (2a-2i, dvs. ni oppgaver) *nøye*, før du begynner å besvare noe som helst. Studér vedlegget, som inneholder mange viktige opplysninger som du trenger/ skal bruke. Legg merke til `consten`, klassene med datamedlemmer og (*ferdiglagde*) funksjoner (inni/utenfor klassene), global variabel, `main` og `skrivMeny()`. Legg spesielt merke til de tre overloadede `By::skrivData(…)`-funksjonene, og når og hvordan disse skal kalles/lages. Husk også på de ferdiglagde funksjonene på `LesData2.h`. Bruk *alt* dette *svært* aktivt.

Vi skal lage et program som holder orden på unike (i hvert fall i byen der den ligger) restauranter i ulike unike byer. Det skal skrives ut ulike oversikter for byene/restaurantene.

Datastrukturen

Datastrukturen består kun av *vectoren* `gByene`. Alle byene har et *unikt* navn/ID, og alle restauranter i *en og samme* by har *unike* navn (likenavnede må gjerne forekomme i ulike byer, f.eks. Egon eller Peppes Pizza). Legg merke til `map`'en med `Restaurant`'er inni klassen `By`.

Oppgaven

a) **Skriv innmaten til** `int finnBy(const string nvn)`

Funksjonen prøver å finne en by med navnet `nvn` og returnerer dets indeks (0 (null) og oppover) i den globale vectoren. Er det ingen by med dette navnet, returneres -1 (minus en).

b) **Skriv innmaten til** `void skrivAlle()` **og** `void By::skrivData()`

Funksjonene sørger *til sammen* for at følgende skrives ut på skjermen: *alle* byenes navn, deres land, antall restauranter i byen, samt *kun* navnet på *alle* restaurantene i vedkommende by.

c) **Skriv innmaten til** `void nyBy()` **og** `void By::lesData()`

Den første funksjonen leser først et bynavn. Finnes denne fra før (bruk funksjonen fra oppg.2a), kommer det en egen melding. I motsatt fall opprettes en ny by, *kun* dens land leses inn (vha. den andre funksjonen), og den legges så til slutt inn bakerst i datastrukturen.

d) **Skriv innmaten til** `void nyRestaurant()`, `void By::leggInnNyRestaurant()` **og** `void Restaurant::lesData()`

Den første funksjonen leser først et bynavn. Finnes denne *ikke*, kommer det en egen melding. I motsatt kalles den andre funksjonen. Denne spør om et restaurantnavn. Finnes denne allerede i vedkommende by, kommer det også en egen melding. I motsatt fall opprettes en ny restaurant, *alle* dens data (unntatt `antallBesok` som nullstilles) leses inn (vha. den tredje funksjonen), og den legges så til slutt inn i datastrukturen.

e) **Skriv innmaten til** `void skrivAltIEnBy()`, `void By::skrivAlt()` **og** `void Restaurant::skrivData()`

Den første funksjonen gjør eksakt det samme som den første forrige deloppgave, bare at en annen andre funksjon kalles. Denne andre funksjonen sørger for at *alle* byens datamedlemmer, samt *alt* om *alle* restaurantene (jfr. bl.a. den tredje funksjonen) i byen skrives ut på skjermen.

f) **Skriv innmaten til** `void skrivMedStjerner()` **og**
`void By::skrivData(const int antallS)`
Den første funksjonen ber om et antall stjerner (0 til `MAXSTJERNER`). Deretter gås det gjennom alle byene, og *alle* restauranter i *alle* byer med *minimum* dette antall stjerner skrives ut med *alle* sine data på skjermen vha. den andre funksjonen (som da også bl.a. bruker en ferdiglaget (i vedlegget) funksjon, samt den tredje i oppg.2e).

g) **Skriv innmaten til** `void skrivNavngitte()` **og**
`void By::skrivData(const string nvn)`
Den første funksjonen ber om (deler av) et restaurantnavn. Deretter gås det gjennom alle byene, og *alle* restauranter i *alle* byer som *inneholder* `nvn` i navnet sitt skrives ut eksakt på samme måte som i forrige deloppgave. For hver by skrives til slutt *antall* restauranter som matchet den aktuelle teksten. **NB:** Husk altså at `nvn` kan hende bare er en sub-/deltekst av restaurantnavnet.

h) **Skriv innmaten til** `void lesFraFil()` **og constructurene med `inn` som parametre**
Funksjonen og constructorene sørger til sammen for at *alle* data om byene og deres restauranter leses inn fra filen «RESTAURANTER.DTA». Aller først på filen ligger antall byer på filen. Formatet for resten av fila bestemmer du helt selv, men **dette *skal* oppgis som en del av besvarelsen**. La vectoren med byer til slutt bli sortert (på bynavnet).

i) **Skriv innmaten til** `void slettAlt()` **og** `By::~By()`
Funksjonen og destructoren sørger til sammen for at *alle* allokerte data om byer, restauranter og pekerne til disse blir slettet fra hukommelsen/memory.

Annet (klargjørende):

- Det er *ikke* en del av denne eksamensoppgaven f.eks. å lage kode som:
  - skriver hele datastrukturen til fil
  - registrere (enda) ett besøk i en restaurant (dvs. øke `antallBesok`)
  - endrer mange av medlemsdataene (som navn, land, adresse, telefon, antStjerner, antallPlasser)

- 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 og bon appétit på neste (Michelin-)restaurantbesøk!

FrodeH

Vedlegg til PROG1003, august 2022: Halvferdig programkode

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


const int MAXSTJERNER = 3;       ///<  Maksimum antall Michelin-stjerner.

/**
 *  Restaurant (med navn ('first' i map'en), adresse, telefon og
 *              antall av: Michelin-stjerner, bord-/sitteplasser og besøk).
 */
class Restaurant  {
  private:                       //  navn = 'first' ifm. map'en.
    string  adresse;             //  Bl.a. IKKE med: hjemmeside,  mail-adresse
    int     telefon, antStjerner,
            antallPlasser, antallBesok;
  public:
    Restaurant()  { telefon = antStjerner = antallPlasser = antallBesok = 0; }
    Restaurant(ifstream & inn);                //  Oppgave 2H
    int  hentAntStjerner()  const  {  return antStjerner;  }
    void lesData();                            //  Oppgave 2D
    void skrivData()  const;                   //  Oppgave 2E
};


/**
 *  By (med navn, land og (Michelin-)restaurantene i byen).
 */
class By  {
  private:
    string navn, land;
    map <string, Restaurant*> restaurantene;
  public:
    By(const string nvn)  { navn = nvn;  }
    By(ifstream & inn);                        //  Oppgave 2H
    ~By();                                     //  Oppgave 2I
    string hentID()   const  {  return navn;  }
    void leggInnNyRestaurant();                //  Oppgave 2D
    void lesData();                            //  Oppgave 2C
    void skrivAlt()   const;                   //  Oppgave 2E
    void skrivData()  const;                   //  Oppgave 2B
    void skrivData(const int antallS) const;   //  Oppgave 2F
    void skrivData(const string nvn)  const;   //  Oppgave 2G
};

int  finnBy(const string nvn);                 //  Oppgave 2A
void lesFraFil();                              //  Oppgave 2H
void nyBy();                                   //  Oppgave 2C
void nyRestaurant();                           //  Oppgave 2D
void skrivAlle();                              //  Oppgave 2B
void skrivAltIEnBy();                          //  Oppgave 2E
void skrivMedStjerner();                       //  Oppgave 2F
void skrivNavngitte();                         //  Oppgave 2G
void skrivMeny();
void slettAlt();                               //  Oppgave 2I


vector <By*> gByene;              ///<  HELE datastrukturen med ALLE byene.
```

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

  lesFraFil();                                    //  Oppgave 2H

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

  while (valg != 'Q') {
     switch (valg) {
       case 'A': skrivAlle();             break;  //  Oppgave 2B
       case 'B': nyBy();                  break;  //  Oppgave 2C
       case 'R': nyRestaurant();          break;  //  Oppgave 2D
       case 'E': skrivAltIEnBy();         break;  //  Oppgave 2E
       case 'S': skrivMedStjerner();      break;  //  Oppgave 2F
       case 'N': skrivNavngitte();        break;  //  Oppgave 2G
       default:  skrivMeny();             break;
     }
     valg = lesChar("\nKommando");
  }

  slettAlt();                                     //  Oppgave 2I

  return 0;
}

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

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


/**
 *  Oppgave 2D - Inn ALLE egne datamedlemmer.
 */
void Restaurant::lesData()  {                              /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2E - Skriver ALLE data ut på skjermen om restauranten.
 */
void Restaurant::skrivData() const  {                      /*   LAG INNMATEN   */  }

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

/**
 *  Oppgave 2H -  Leser inn ALLE egne data fra fil.
 *
 *  @param   inn  -  Filobjektet det leses inn data fra
 *  @see     Restaurant::Restaurant(...)
 */
By::By(ifstream & inn)  {                                  /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2I -  Sletter alle eksisterende/allokerte restauranter.
 */
By::~By()  {                                               /*   LAG INNMATEN   */  }
```

```
/**
 *  Oppgave 2D -Legger inn (om mulig) en ny restaurant i byen.
 *  @see   Restaurant::Restaurant()
 *  @see   Restaurant::lesData()
 */
void By::leggInnNyRestaurant()  {                          /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2C - Leser KUN inn hvilket land byen ligger i.
 */
void By::lesData()  {                                      /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2E - Skriver ALT om ALLE restaurantene i byen.
 *  @see   Restaurant::skrivData()
 */
void By::skrivAlt() const  {                               /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2B - Skriver alle byens HOVEDdata.
 */
void By::skrivData()  const  {                             /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2F - Skriver (om mulig) alle restauranter med
 *               et MINIMUM av Michelin-stjerner.
 *  @param    antallS  -  Minimum antall stjerner i restaurantene
 *  @see      Restaurant::hentAntStjerner()
 *  @see      Restaurant::skrivData(...)
 */
void By::skrivData(const int antallS) const  {             /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2G - Skriver (om mulig) alle restauranter med et gitt (del)navn.
 *  @param    nvn  -  Aktuelt (del)navn på restaurant
 *  @see      Restaurant::skrivData(...)
 */
void By::skrivData(const string nvn)  const  {             /*   LAG INNMATEN   */  }


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

/**
 *  Oppgave 2A - Leter etter gitt bynavn og returnerer om mulig dets indeks.
 *  @param    nvn  -  Aktuelt bynavn å finne/lete etter.
 *  @return   Indeksen for 'nvn' i 'gByene', evt. -1 (minus en) om intet funn.
 *  @see      By::hentID()
 */
int finnBy(const string nvn)  {                            /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2H - Leser ALLE byene og restaurantene inn fra fil.
 *  @see   By::By(...)
 *  @see   By::hentID()
 */
void lesFraFil()  {                                        /*   LAG INNMATEN   */  }
```

```
/**
 *  Oppgave 2C - Legger inn (om mulig) en ny by.
 *  @see  By::By(...)
 *  @see  By::lesData()
 */
void nyBy()  {                                          /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2D - Legger inn (om mulig) ny restaurant i en eksisterende by.
 *  @see   finnBy(...)
 *  @see   By::leggInnNyRestaurant()
 */
void nyRestaurant()  {                                  /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2B - Skriver ALLE byenes hoveddata.
 *  @see   By::skrivData()
 */
void skrivAlle()  {                                     /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2E - Skriver (om mulig) ALT om ALLE restaurant i en gitt by.
 *  @see   finnBy(...)
 *  @see   By::skrivAlt()
 */
void skrivAltIEnBy()  {                                 /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2F - Skriver ALLE restauranter i ALLE byer
 *               med et visst antall MINIMUM Michelin-stjerner.
 *  @see   By::skrivData(...)
 */
void skrivMedStjerner()  {                              /*   LAG INNMATEN   */  }


/**
 *  Oppgave 2G - Skriver ALLE restauranter i ALLE byer med et gitt (del)navn.
 *  @see   By::skrivData(...)
 */
void skrivNavngitte()  {                                /*   LAG INNMATEN   */  }


/**
 *  Skriver programmets menyvalg/muligheter på skjermen.
 */
void skrivMeny()  {
  cout << "\nFølgende kommandoer er tilgjengelige:\n"
       << "   A  - skriver Alle byene og navnene på Alle restaurantene i hver\n"
       << "   B  - ny By\n"
       << "   R  - ny Restaurant (i en gitt by)\n"
       << "   E  - skriv alt om alle restaurantene En gitt by\n"
       << "   S  - skriv alle restauranter med et minimum antall Stjerner\n"
       << "   N  - skriv alle restauranter med en gitt tekst i Navnet\n"
       << "   Q  - Quit / avslutt\n";
}


/**
 *  Oppgave 2I - Sletter HELE datastrukturen fra memory.
 *  @see   By::~By()
 */
void slettAlt()  {                                      /*   LAG INNMATEN   */  }
```