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 2023**
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 (28%)

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

```
#include <iostream>
#include <string>     //  'find' returnerer evt. startindeksen for tekstmatch
using namespace std;


class X  {
  protected:
    string nvn;
    bool   student;

  public:
    X(const string t, const bool s) { nvn = t;  student = s; }
    virtual void display() const { cout << nvn << ' ' << student; }
    void har(const string t) const  {  cout << nvn.find(t);  }
    bool lik(const bool s) const  {  return (student == s);  }
    virtual bool ulik(const string t) const  {  return (nvn != t);  }
};


class Y : public X {
  private:
    int  alder;

  public:
    Y(const string t, const bool s, const int a) : X(t, s)  {  alder = a;  }
    virtual void display() const  {  X::display();  cout << ", " << alder;  }
    bool lik(const string t) const  {  return (nvn == t);  }
    virtual bool ulik(const int a) const  {  return (alder != a);  }
};


int main() {
  Y* student1 = new Y("Iris", true, 20);
  X* student2 = new X("Arne", false);
  Y* student3 = new Y("Mari", true, 21);

  student3->display();  cout << "   ";  student2->display();   cout << '\n';
  cout << student1->ulik(21) << "   " << student2->ulik("Arna")     << '\n';
  student1->har("is");  cout << "   ";  student3->har("r");    cout << '\n';
  cout << ((student2->lik(true))   ? "AIM" : "MIA") << '\n';
  cout << ((student1->lik("Irma")) ? "MAI" : "MIA") << '\n';

  delete student1;  delete student2;  delete student3;
  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 <char> tegn1(6, '*');
    vector <char> tegn2(2);
    vector <char> tegn3 { 'K', 'A', 'R', 'K', 'I', 'N', 'A', 'G', 'R', 'I' };
    vector <char> tegn4(tegn1);

    tegn2 = tegn4;
    for (const auto & val : tegn2) cout << ' ' << val;   cout << " : ";
    for (const auto & val : tegn3) cout << ' ' << val;   cout << '\n';

    cout << tegn3[3] << ' ' << tegn3.back() << ' ' << tegn3.at(5) << ' '
         << tegn2.front() << ' ' << tegn2[tegn2.size()-4]     << '\n';

    tegn2 = tegn3;
    auto it  = find(tegn2.begin(),  tegn2.end(),  'R');
    auto it2 = find(tegn2.rbegin(), tegn2.rend(), 'R');

    it++;  it2++;
    while (*it != *it2) cout << *it++ << ' ';            cout << '\n';


    map <int, string> navn;
    navn[18] = "Johan";
    navn.insert(pair <int, string>(29, "Augstus"));
    navn[7]  = "Simen";
    navn.insert(pair <int, string>(17, "May"));
    navn[16]  = "Andre";
    navn.insert(pair <int, string>(22, "April"));

    for (const auto & val : navn)
        cout << val.second << "  ";                      cout << '\n';

    for (auto it = navn.find(18);  it != navn.end();  it++)
        cout << it->first << ' ';                        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 spesielt merke til klassene med datamedlemmer og (*ferdiglagde*) funksjoner (utenfor/ inni klassene), og at de er både `private`, `protected` og `public`), global variabel, `main` og `skrivMeny`. Husk også på funksjonene på `LesData2.h`. Bruk *alt* dette *svært* aktivt.

Professor Pierre fra Paris har bosatt seg i Alba i Piemonte. Som professor og single, så har Pierre god økonomi. Derfor går han *veldig* ofte ut og spiser, og ikke sjeldent spiser Pierre både lunsj og middag ute samme dag. Men han går *aldri* og spiser på samme sted to ganger den samme dagen. Pierre er også veldig prinsippfast og rutinepreget. Derfor spiser han *kun en* hovedrett til lunsj, drikker vann, og evt. kaffe *bare* da (*ikke* til middag). Til middag kan han spise *opptil* tre retter (forrett, hovedrett, dessert). Han spiser *alltid minst en* av disse. I tillegg *kan* han også ta noe drikke (vann, brus, farris, øl eller vin). For alt dette siste: se klassene og kommentarer i vedlegget.
Vi skal lage et program som holder orden *alle* Pierres ulike måltider på ulike spisesteder i Piemonte i løpet av *ett og samme* kalenderår. Derfor er datoene på formatet: MMDD (måned-dag).

Datastrukturen

Datastrukturen består *kun* av *listen* `gSpisestedene`. Alle spisestedene har et *unikt* navn.
Hvert spisested inneholder bl.a. en `map` med *alle* måltidene (lunsjer og/eller middager) spist på stedet.
Alle datoene i denne `map`'en er altså unike (da han altså *aldri* spiser samme sted to ganger samme dag).

Oppgaven

a) 8 Skriv innmaten til

```
void skrivAlleSpisesteder()
void Spisested::skrivHovedData()
```

Det kommer en egen melding om ingen spisesteder finnes. Ellers skrives antall spisesteder ut på skjermen, samt *hoved*dataene om hvert enkelt spisested. Dette siste gjøres vha. den andre funksjonen, som skriver ut *alle* klassens string-data, samt *kun antallet* av måltider på spisestedet.

b) 7 Skriv innmaten til

```
Spisested* finnSpisested(const string navn)
```

Leter i listen etter et spisested med navnet `navn`. Lykkes dette returneres en peker til det aktuelle spisestedet, ellers returneres `nullptr`.

c) 8 Skriv innmaten til

```
void skrivEttSpisested()
void Spisested::skrivAlleData()
void Lunsj::skrivData()
void Middag::skrivData()
```

Det kommer en egen melding om ingen spisesteder finnes. Ellers leses et spisesteds navn.
Er dette *ikke* å finne (jfr. oppgave 2b) kommer en melding. I motsatt fall skrives *alle* stedets data ut vha. den andre funksjonen. Denne funksjonen skriver stedets *hoved*data (jfr. oppgave 2a).
Om det er registrert måltider på stedet, skrives også *alle* data om *alle* disse ut på skjermen (vha. de to siste funksjonene). `string`'er skrives (selvsagt) ut som tekster.
Er en tekst blank/tom, skrives *ingenting* om den. `bool` skrives ut som en passende tekst.
Datoene *skal* skrives på formen: DD / MM

d) 8 **Skriv innmaten til** `void nyttSpisested()` og `void Spisested::lesData()`
Et nytt spisesteds navn leses. Finnes dette allerede, komme det en egen melding. I motsatt fall opprettes et nytt spisested og dets to andre `string`-data leses inn (vha. den andre funksjonen). Den legges inn bakerst i datastrukturen, og til slutt resorteres listen.
NB: Måltider skal *ikke* leses inn i den andre funksjonen. Det gjøres i oppgave 2f.

e) 6 **Skriv innmaten til** `Maaltid* Spisested::finnMaaltidPaaDato(const int dato)`
Leter i `map`'en etter et måltid på `dato`. Lykkes dette returneres en peker til det aktuelle måltidet, ellers returneres `nullptr`.

f) 11 **Skriv innmaten til** `void nyttMaaltid()`, `void Spisested::nyttMaaltid()`, `void Lunsj::lesData()` **og** `void Middag::lesData()`
Det kommer en egen melding om ingen spisesteder finnes. Ellers leses et spisesteds navn. Er dette å finne (jfr. oppgave 2b) leses et nytt måltid på stedet inn fra brukeren vha. den andre funksjonen. Denne funksjonen leser inn måltidets dag (DD) og måned (MM). Finnes det allerede et måltid på denne datoen (på dette stedet), kommer det en melding. Eller spørres, *og sikres*, det om dette er L(unsj) eller M(iddag). Aktuelt objekt opprettes, alle dets data leses inn (vha. de to siste funksjonene), og objektet legges til slutt inn i datastrukturen. Ifm. lunsj leses en ikke-blank tekst for `hovedrett`, samt om `kaffe` er drukket eller ei. Ifm. middag leses både `drikke`, `forrett`, `hovedrett` og `dessert`. *Minst en* av de tre siste *må* være ikke-blank/tom.

g) 8 **Skriv innmaten til** `void spistEnGittDato()`
Skriv *all* koden som trengs for å skrive ut *alle* måltider spist på en gitt innlest dato. Aktuelle spisested(er) skal også skrives ut. Om *ingen* måltider er å finne, kommer det en egen melding.

h) 6 **Skriv innmaten til** `void frigiAllokertMinne()` **og** `Spisested::~Spisested()`
Slett *alle* objekter i *hele* listen, og alle pekerne til dem, samt for hvert av dem: alt i `map`'en.

i) 10 **Skriv innmaten til** `void lesFraFil()` **og**
alle de fire constructorene med `inn` som parameter
Disse funksjonene sørger til sammen for at *hele* listen med spisesteder og deres måltider leses inn fra filen «SPISESTEDER.DTA». Aller først på filen ligger antall spisesteder. Formatet for resten av filen bestemmer du helt selv, men **dette *skal* oppgis som en del av besvarelsen**.
NB: Listen skal være sortert på `navn` etterpå (for vi har ingen garanti for at den er det på filen).

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).

Bon appetit - velbekomme!

FrodeH

Vedlegg til PROG1003, 16.mai 2023: Halvferdig programkode

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

class Maaltid;                     //  Pre-deklarasjon.

/**
 *  Spisested (navn, adresse, telefon og ALLE måltidene spist der).
 */
class Spisested {
  private:
    string navn,                     //  UNIKT navn - sortert på dette.
           adresse,
           tlf;                      //  'tlf' er string, da også KAN
                                     //    inneholde '+', landskode og blanke.
    map <int, Maaltid*> maaltidene;  //  'int' er dato på formatet: MMDD
                                     //    og er UNIKT for hvert spisested.
  public:
    Spisested(const string nvn)  {  navn = nvn;  }      //       (Ferdiglaget)
    Spisested(ifstream & inn);                          //  Oppgave 2I
    ~Spisested();                                       //  Oppgave 2H
    Maaltid* finnMaaltidPaaDato(const int dato) const;  //  Oppgave 2E
    string hentID() const  { return navn; }             //       (Ferdiglaget)
    void lesData();                                     //  Oppgave 2D
    void nyttMaaltid();                                 //  Oppgave 2F
    void skrivAlleData()  const;                        //  Oppgave 2C
    void skrivHovedData() const;                        //  Oppgave 2A
};

/**
 *  Baseklassen Maaltid (kun med navnet på måltidets hovedrett).
 */
class Maaltid  {
  protected:
    string hovedrett;          //  Tekst (lunsj/middag) eller blank (middag).

  public:
    Maaltid()  {  }
    Maaltid(ifstream & inn);                            //  Oppgave 2I
    virtual ~Maaltid()  {  }
    virtual void lesData() = 0;              //  = 0 betyr 'pure virtual',
    virtual void skrivData() const = 0;      //    dvs. subklasser MÅ lage dem.
};

/**
 *  Avledet klasse Lunsj (kun med om har drukket kaffe eller ei til lunsj).
 */
class Lunsj : public Maaltid  {
  private:
    bool kaffe;                //  Drikker evt. KUN vann i tillegg (til lunsj).

  public:
    Lunsj()  {  }
    Lunsj(ifstream & inn);                              //  Oppgave 2I
    virtual ~Lunsj()  {  }
    virtual void lesData();                             //  Oppgave 2F
    virtual void skrivData() const;                     //  Oppgave 2C
};
```

```
/**
 *  Avledet klasse Middag (med evt navn på forrett, dessert og/eller drikken).
 */
class Middag: public Maaltid  {
  private:
    string forrett,           //  Tekst eller blank.
           dessert,           //  Tekst eller blank.
           drikke;            //  Tekst eller blank.

  public:
    Middag()  {  }
    Middag(ifstream & inn);                              //  Oppgave 2I
    virtual ~Middag()  {  }
    virtual void lesData();                              //  Oppgave 2F
    virtual void skrivData() const;                      //  Oppgave 2C
};


Spisested* finnSpisested(const string navn);  //  Oppgave 2B
void frigiAllokertMinne();                    //  Oppgave 2H
void lesFraFil();                             //  Oppgave 2I
void nyttMaaltid();                           //  Oppgave 2F
void nyttSpisested();                         //  Oppgave 2D
void skrivAlleSpisesteder();                  //  Oppgave 2A
void skrivEttSpisested();                     //  Oppgave 2C
void skrivMeny();
void spistEnGittDato();                       //  Oppgave 2G


list <Spisested*> gSpisestedene;              ///<  ALLE aktuelle spisesteder.


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

  lesFraFil();                                        //  Oppgave 2I

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

  while (valg != 'Q') {
     switch (valg) {
       case 'A': skrivAlleSpisesteder();   break;  //  Oppgave 2A
       case 'E': skrivEttSpisested();      break;  //  Oppgave 2C
       case 'S': nyttSpisested();          break;  //  Oppgave 2D
       case 'M': nyttMaaltid();            break;  //  Oppgave 2F
       case 'D': spistEnGittDato();        break;  //  Oppgave 2G
       default:  skrivMeny();              break;
     }
     valg = lesChar("\nKommando");
  }

  frigiAllokertMinne();                               //  Oppgave 2H

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

```
/** Skriver programmets menyvalg/muligheter på skjermen.
 */
void skrivMeny()  {
  cout << "\nFølgende kommandoer er tilgjengelige:\n"
       << "   A  - skriv Alle spisesteder\n"
       << "   E  - skriv Ett spisested\n"
       << "   S  - nytt Spisested\n"
       << "   M  - nytt Maaltid\n"
       << "   D  - steder spist paa en gitt Dato\n"
       << "   Q  - Quit / avslutt\n";
}


/** Oppgave 2I - Leser inn fra fil ALT om ETT spisested.
 *  @param    inn  -  Filen det leses inn fra
 */
Spisested::Spisested(ifstream & inn)  {              /* LAG INNMATEN  */  }


/** Oppgave 2H - Sletter/fjerner ALLE data inni ETT spisested.
 */
Spisested::~Spisested()  {                           /* LAG INNMATEN  */  }


/** Oppgave 2E - Finner og returnerer (om mulig) 'Maaltid' på en gitt dato.
 *  @param    dato  -  Datoen det ønskes å finne et måltid på
 *  @return   Måltidet på gitt dato, evt. nullptr (om ingen dato-match)
 */
Maaltid* Spisested::finnMaaltidPaaDato(const int dato) const  { /*LAG INNMATEN */ }


/** Oppgave 2D - Leser inn spisestedets resterende enkelt-datamedlemmer.
 */
void Spisested::lesData()  {                         /* LAG INNMATEN  */  }


/** Oppgave 2F - Legger inn (om mulig) nytt måltid på en ny dato.
 */
void Spisested::nyttMaaltid()  {                     /* LAG INNMATEN  */  }


/** Oppgave 2C - Skriver ut på skjermen ALLE data om ETT spisested.
 */
void Spisested::skrivAlleData() const  {             /* LAG INNMATEN  */  }


/** Oppgave 2A - Skriver ut på skjermen enkelt-datamedlemmene.
 */
void Spisested::skrivHovedData() const  {            /* LAG INNMATEN  */  }


/** Oppgave 2I - Leser inn 'Maaltid'ets eneste datamedlem.
 *  @param    inn  -  Filen det leses inn fra
 */
Maaltid::Maaltid(ifstream & inn)  {                  /* LAG INNMATEN  */  }


/** Oppgave 2I - Leser inn fra fil 'Lunsj' sitt eneste datamedlem.
 *  @param    inn  -  Filen det leses inn fra
 */
Lunsj::Lunsj(ifstream & inn) : Maaltid(inn)  {       /* LAG INNMATEN  */  }


/** Oppgave 2F - Leser inn (hoved)retten under lunsjen.
 */
void Lunsj::lesData()  {                             /* LAG INNMATEN  */  }
```

```
/** Oppgave 2C - Skriver ut på skjermen ALLE datamedlemmene.
 */
void Lunsj::skrivData() const  {                        /* LAG INNMATEN  */  }


/** Oppgave 2I - Leser inn fra fil 'Middag' sine tre datamedlem.
 *  @param    inn  -  Filen det leses inn fra
 */
Middag::Middag(ifstream & inn) : Maaltid(inn)  {     /* LAG INNMATEN  */  }


/** Oppgave 2F - Leser inn evt rettene og drikken under middagen.
 */
void Middag::lesData()  {                               /* LAG INNMATEN  */  }


/** Oppgave 2C - Skriver ut på skjermen ALLE datamedlemmene.
 */
void Middag::skrivData() const  {                       /* LAG INNMATEN  */  }


/** Oppgave 2B- Finner og returnerer (om mulig) peker til et gitt spisested.
 *  @param    navn  -  Navnet på det søkte spisestedet
 *  @return   Peker til aktuelt navngitt spisested eller nullptr
 */
Spisested* finnSpisested(const string navn)  {       /* LAG INNMATEN  */  }


/** Oppgave 2H - Frigir/sletter ALT allokert minne/memory.
 */
void frigiAllokertMinne()  {                            /* LAG INNMATEN  */  }


/** Oppgave 2I - Leser ALLE spisestedene med måltider inn fra fil.
 */
void lesFraFil()  {                                     /* LAG INNMATEN  */  }


/** Oppgave 2F - Legger inn (om mulig) ett nytt måltid på et spisested.
 */
void nyttMaaltid()  {                                   /* LAG INNMATEN  */  }


/** Oppgave 2D - Legger inn (om mulig) ett nytt spisested.
 */
void nyttSpisested()  {                                 /* LAG INNMATEN  */  }


/** Oppgave 2A - Skriver ut på skjermen HOVEDdataene om ALLE spisesteder.
 */
void skrivAlleSpisesteder()  {                          /* LAG INNMATEN  */  }


/** Oppgave 2C - Skriver ut på skjermen ALLE data om ETT gitt spisested.
 */
void skrivEttSpisested()  {                             /* LAG INNMATEN  */  }


/** Oppgave 2G - Skriver ut på skjermen (om mulig) alle måltider spist en
 *               gitt dato, og på hvilket spisested dette har forekommet.
 */
void spistEnGittDato()  {                               /* LAG INNMATEN  */  }
```