NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamensoppgave i PROG1003
– Objekt-orientert programmering

Faglig kontakt under eksamen:	**Frode Haug**
Eksamensdato:	**20.mai 2025**
Eksamenstid (fra-til):	**15:00-19:00 (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 <vector>
#include <cctype>
using namespace std;

class A  {
  protected:
    vector <string> navnene;
  public:
    virtual void funk1(const string nvn)  {  navnene.push_back(nvn);  }
    virtual void funk2()  {
      for (auto const & val : navnene) cout << val << '\n';
    }
    void funk3()  {  cout << navnene.size();  }
    int  funk4(const char ch)  {
      int ant = 0;
      for (int i = 0;  i < navnene.size();  i++)
          if (navnene[i].find(toupper(ch))  !=  string::npos  ||
              navnene[i].find(tolower(ch))  !=  string::npos) ant++;
      return ant;
    }
};

class B : public A {
  private:
    vector <int> moh;
  public:
    void funk1(const string nvn, const int moh)  {
      A::funk1(nvn);   (this->moh).push_back(moh);
    }
    void funk2()  {
      for (int i = 0;  i < moh.size();  i++)
          cout << navnene[i] << '-' << moh[i] << "   ";
    }
    void funk3()  {  cout << "Antall: ";  A::funk3();  }
    int  funk5(const int h1, const int h2)  {
      int ant = 0;
      for (auto const & val : moh)  if (val >= h1  &&  val <= h2) ant++;
      return ant;
    }
};

int main() {
  A* objA;
  B* objB = new B;
  objB->funk1("Mor", 345); objB->funk1("Ark", 123);
  objB->funk1("Far", 234);        (*objB).funk2(); cout << '\n';
  objA = objB;                    objA->funk2();   cout << '\n';
  objB->funk1("Tur", 678); objB->funk1("Par", 567);  objB->funk1("VAR", 789);
  objA->funk3();  cout << "   ";  objB->funk3();   cout << '\n';
  cout << objB->funk5(456, 765)                          << '\n';
  cout << objB->funk4('R') << " + " << objB->funk4('a') << '\n';
  return 0;
}
```

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

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


int main()  {
  map <int, char> m;
  queue <char> q;
  int antall = 0;
  string str;
  char ch;

  m[17] = 'N';  m[29] = 'L';  m[22] = 'A';  m[11] = 'S';
  m[7]  = 'R';  m[15] = 'E';  m[6]  = 'A';


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


  auto it = m.rbegin();
  while (it != m.rend())  q.push((it++)->second);
  for (int i = 0;  i < q.size();  i++)  {
      cout << q.front() << ' ';
      q.push(q.front());  q.pop();
  }
  cout << '\n';


  for (int i = 0;  i < 4;  i++)  {
      str += q.front();   q.pop();
  }
  cout << str << '\n';


  auto it2 = m.begin();   advance(it2, 4);
  m.erase(it2, m.end());
  it2 = m.begin();
  while (it2 != m.end())
    cout << (it2++)->second;
  cout << '\n';


  for (int i = 0;  i < str.length();  i++)
      for (auto const & val : m)
          if (val.second == str[i]) antall++;
  cout << antall << '\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, alle klassene med arv, datamedlemmer og (*ferdiglagde*) funksjoner inni klassene, global variabel, `main` og `skrivMeny`. Legg ekstra spesielt merke til i `main` hvordan funksjoner inni `Kulturhus` kalles direkte ifm. oppgavene 2B-2E. Husk også på funksjonene på `LesData2.h`. Bruk *alt* dette *svært* aktivt.

Det skal holdes orden på ulike arrangement (konserter og teater-forestillinger) i ulike kulturhus.

Datastrukturen

Datastrukturen består *kun* av *vectoren* `gKulturhusene`. Hvert kulturhus inneholder bl.a. en `list` med de ulike arrangementene (konserter og teater-forestillinger) som foregår i vedkommende kulturhus.
NB: `type` inni `Arrangement` brukes *kun* ifm. oppgave 2F og 2G.

Oppgaven

a) 8 Skriv innmaten til

```
void nyttKulturhus()
void Kulturhus::lesData()
```

Disse funksjonene sørger til sammen for at et nytt `Kulturhus` opprettes, *alle* (unntatt arrangementer) dets data leses inn, og det legges inn i datastrukturen.
NB: Det er ingen sjekk på om duplikate kulturhus forekommer.

b) 10 Skriv innmaten til

```
void Kulturhus::skrivArrangement()
```

og de tre virtuelle funksjonene

```
void skrivData()
```

Den første funksjonen skriver først ut kulturhusets to datamedlemmer. Finnes det ingen arrangementer der, kommer det en egen melding. I motsatt fall skrives (vha. de tre virtuelle funksjonene) *alle* arrangementer med *alle* sine data ut på skjermen.

c) 13 Skriv innmaten til

```
void Kulturhus::nyttArrangement()
```

og de tre virtuelle funksjonene

```
void lesData()
```

Den første funksjonen spør først om, *og sikrer*, om brukeren ønsker et arrangement av typen K(onsert) eller T(eater). Aktuelt objekt opprettes, *alle* dets data leses inn (vha. de virtuelle funksjonene) og det legges inn i datastruktuen. Til slutt sørges det for at arrangementene på kulturhuset hele tiden holdes sortert på stigende dato (som er på formen ÅÅMMDD).
NB: Det er ingen sjekk på om duplikate arrangementer forekommer.

d) 6 Skriv innmaten til

```
void Kulturhus::skrivArrangement2()
```

Lag en ny versjon av den første funksjonen i 2B, bare at brukeren spørres innledningsvis om to datoer (der den siste *må* være større eller lik den første). Arrangementer *kun* fra om med den første datoen og opp til og med den siste datoen skrives ut på skjermen.

e) 12 **Skriv innmaten til** `void Kulturhus::fjernArrangement()`
Aller først og aller sist i funksjonen skrives alle nåværende arrangementer på kulturhuset ut på skjermen. Mellom dette spørres det om en dato. Det gås så igjennom alle arrangementer på kulturhuset. For hvert arrangement på den aktuelle datoen (for det *kan* være flere), så skrives arrangementet ut på skjermen. Det spørres om vedkommende skal slettes/fjernes. Er dette ønskelig, så skjer det. **NB:** Det *skal* brukes `iterator` for å gå igjennom listen.

f) 11 **Skriv innmaten til** `void skrivAltPaaGittDato()` **og all annen kode som trengs** for å få skrevet ut *alle* arrangementer på *alle* kulturhus som forekommer på en ønsket dato. Brukeren skal også kunne velge om vedkommende ønsker å ha skrevet ute begge typer arrangementer på den aktuelle datoen, eller bare en av typene (konsert eller teater).
NB1: Her får du bruk for å få tak i og bruke arrangementstypen (`type`).
NB2: Det *skal* bl.a. brukes `for_each(.....)` og lambda-funksjon til dette.

g) 10 **Skriv innmaten til** `void lesFraFil()` **og alle de fire constructorene med `inn` som parameter**
Disse funksjonene sørger til sammen for at *hele* datastrukturen leses inn fra filen «KULTURHUS.DTA». På filen *skal* bl.a. antall kulturhus ligge/være. Formatet på filen bestemmer du ellers helt selv, men **det *skal* oppgis som en del av besvarelsen**.
NB1: Her må nok bl.a. `type` oppdateres korrekt i constructorene.
NB2: Listen inni hvert arrangement *skal* være sortert på stigende dato etterpå (for vi har ingen garanti for at den er det på filen).

Annet (klargjørende):

- dato er altså på formen ÅÅMMDD og skal ved innlesning fra brukeren stort sett ligge mellom FDATO (250101) og SDATO (301231). Men det er ingen sjekk på dets gyldighet utover dette. Altså kan vi risikere at brukeren angir datoen: 264589 – dvs. 89/45-26. Det får vi bare leve med …
- Som arrangementsted er det altså brukt begrepet «Kulturhus». Dette kan selvsagt være et annet sted også, f.eks slikt som: samfunnshus, kirke, konserthus, opera, (ute)scene, …..
- Det er *kun* arrangementer av typen konsert og teater-forestilling som er aktuelt. Alle andre typer (som f.eks. dans, opera, foredrag) er det ikke aktuelt å legge inn via programmet vårt.
- 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 (eksamens)kulturell opplevelse!

FrodeH

Vedlegg til PROG1003, 20.mai 2025: Halvferdig programkode

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

const int FDATO  = 250101;    ///<  Første mulige dato (1/1-2025).
const int SDATO  = 301231;    ///<  Siste mulige dato (31/12-2030).
const int FKL    =    600;    ///<  Første mulige klokkeslett for arrangement.
const int SKL    =   2200;    ///<  Siste mulige klokkeslett for arrangement.
const int MINAKT =      1;    ///<  Minimum antall aktører i teaterstykke.
const int MAXAKT =    300;    ///<  Maksimum antall aktører i teaterstykke.

/**
 *  Baseklassen 'Arrangement' med dets navn, sal/rom/lokasjon det skjer i,
 *    samt datoen og klokkeslettet for arrangementet, i tillegg er det en
 *    'type' som er 'A', 'K' eller 'T' ut fra hvilken type objektet er av
 *    (denne brukes primært ifm oppgave 2F og 2G).
 */
class Arrangement  {
  private:
    string navn, sal;
    int    dato,                                  //  Er på formen:  ÅÅMMDD
           klokkeslett;                           //  Er på formen:  TTMM
  protected:
    char   type;
  public:
    Arrangement()  { type = 'A';  dato = FDATO;  klokkeslett = 0;  }
    Arrangement(ifstream & inn);                                  //  Oppgave 2G
    int  hentID()   const  {  return dato;  }
    char hentType() const  {  return type;  }
    virtual void lesData();                                       //  Oppgave 2C
    virtual void skrivData() const;                               //  Oppgave 2B
};

/**
 *  Avledet klasse 'Konsert' med KUN artistens navn.
 */
class Konsert : public Arrangement  {
  private:
    string artist;
  public:
    Konsert()  { type = 'K';  }
    Konsert(ifstream & inn);                                      //  Oppgave 2G
    virtual void lesData();                                       //  Oppgave 2C
    virtual void skrivData() const;                               //  Oppgave 2B
};

/**
 *  Avledet klasse 'Teater' med regissørens navn og ca.antall aktører.
 */
class Teater : public Arrangement  {
  private:
    string  regissor;
    int     antallAktorer;              // Ca.antall aktører i teaterstykket.
  public:
    Teater()  {  type = 'T';  antallAktorer = 0; }
    Teater(ifstream & inn);                                       //  Oppgave 2G
    virtual void lesData();                                       //  Oppgave 2C
    virtual void skrivData() const;                               //  Oppgave 2B
};
```

```
/**
 *  Klassen 'Kulturhus' med dets navn, beliggenhet (by/sted)
 *    og liste med ALLE nåværende aktuelle arrangementer i kulturhuset.
 */
class Kulturhus  {
  private:
    string navn,                                //  Kulturhusets navn.
           sted;                                //  Stedet/byen det ligger i.
    list <Arrangement*> arrangementene;         //  Alle aktuelle arrangementer.
  public:
    Kulturhus()  {  }
    Kulturhus(ifstream & inn);                                  //  Oppgave 2G
    void fjernArrangement();                                    //  Oppgave 2E
    void lesData();                                             //  Oppgave 2A
    void nyttArrangement();                                     //  Oppgave 2C
    void skrivArrangement()  const;                             //  Oppgave 2B
    void skrivArrangement2() const;                             //  Oppgave 2D
};


void lesFraFil();                                               //  Oppgave 2G
void nyttKulturhus();                                           //  Oppgave 2A
void skrivAltPaaGittDato();                                     //  Oppgave 2F
void skrivMeny();


vector <Kulturhus*> gKulturhusene; ///<  Datastrukturen med ALLE kulturhusene.


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

  lesFraFil();                                                   //  Oppgave 2G

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

  while (valg != 'Q') {
     switch (valg) {
       case 'K': nyttKulturhus();                          break;        //  2A
       case 'A': nr = lesInt("\tNytt i nr",  1, gKulturhusene.size());  //  2C
                 gKulturhusene[nr-1]->nyttArrangement();  break;
       case 'S': nr = lesInt("\tSkriv nr",   1, gKulturhusene.size());  //  2B
                 gKulturhusene[nr-1]->skrivArrangement(); break;
       case 'T': nr = lesInt("\tSkriv nr",   1, gKulturhusene.size());  //  2D
                 gKulturhusene[nr-1]->skrivArrangement2(); break;
       case 'F': nr = lesInt("\tFjern i nr", 1, gKulturhusene.size());  //  2E
                 gKulturhusene[nr-1]->fjernArrangement(); break;
       case 'D': skrivAltPaaGittDato();    break;                       //  2F
       default:  skrivMeny();              break;
     }
     valg = lesChar("\nKommando");
  }

  cout << "\n\n";

  return 0;
}
```

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

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


/**
 *  Oppgave 2C - Leser inn ALLE Arrangementets data.
 */
void Arrangement::lesData()  {                       /*    LAG INNMATEN    */      }


/**
 *  Oppgave 2B - Skriver ut på skjermen ALLE Arrangementets data.
 */
void Arrangement::skrivData() const   {              /*    LAG INNMATEN    */      }

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

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


/**
 *  Oppgave 2C - Leser inn ALLE Konsertens data.
 */
void Konsert::lesData()  {                           /*    LAG INNMATEN    */      }


/**
 *  Oppgave 2B - Skriver ut på skjermen ALLE Konsertens data.
 */
void Konsert::skrivData() const   {                  /*    LAG INNMATEN    */      }

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

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


/**
 *  Oppgave 2C - Leser inn ALLE Teaterets data.
 */
void Teater::lesData()  {                            /*    LAG INNMATEN    */      }


/**
 *  Oppgave 2B - Skriver ut på skjermen ALLE Teatrets data.
 */
void Teater::skrivData() const   {                   /*    LAG INNMATEN    */      }
```

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

/**
 *  Oppgave 2G - ALT om ETT Kulturhus leses inn fra fil.
 */
Kulturhus::Kulturhus(ifstream & inn)  {              /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2E - Fjerner/sletter (om mulig) arrangement(er) på en GITT dato.
 */
void Kulturhus::fjernArrangement()  {                /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2A - Alle Kuturhusets data leses inn (untatt arrangementer).
 */
void Kulturhus::lesData()  {                         /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2C - Nytt arrangement legges inn.
 */
void Kulturhus::nyttArrangement()  {                 /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2B - ALT om ETT Kulturhus skrives ut på skjermen.
 */
void Kulturhus::skrivArrangement()  const {          /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2D - ALT om ETT Kulturhus i en GITT tidsperiode skrives ut.
 */
void Kulturhus::skrivArrangement2()  const {         /*    LAG INNMATEN    */      }


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

/**
 *  Oppgave 2G - Leser ALLE kulturhusene og arrangementene inn fra fil.
 */
void lesFraFil()  {                                  /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2A - Legger inn ett nytt kulturhus.
 */
void nyttKulturhus()  {                              /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2F - Skriver ALLE eller DELER av arrangementer på en GITT dato,
 *               uavhengig av hvilket Kulturhus det foregår på.
 */
void skrivAltPaaGittDato()  {                        /*    LAG INNMATEN    */      }

/**
 *  Skriver programmets menyvalg/muligheter på skjermen.
 */
void skrivMeny()  {
  cout << "\nFolgende kommandoer er tilgjengelige:\n"
       << "   K  - nytt Kulturhus\n"
       << "   A  - nytt Arrangement\n"
       << "   S  - Skriv arrangement i ett kulturhus\n"
       << "   T  - skriv arrangement i ett kulturhus i Tidsperiode\n"
       << "   F  - Fjern arrangement i ett kulturhus\n"
       << "   D  - skriv arrangement paa en gitt Dato\n"
       << "   Q  - Quit / avslutt\n";
}
```