NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamensoppgave i PROG1003
– Objekt-orientert programmering

Faglig kontakt under eksamen:	**Frode Haug**
Eksamensdato:	**4.mai 2026**
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):	**8**

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 <map>
using namespace std;


class A  {
  protected:
    map <string, int> data;

  public:
    A()  {  }

    void funk1(const string t, const int n)  {  data[t] = n;  }

    bool funk2(const string t) { return (data.find(t) != data.end()); }

    virtual int funk3()  {  return data.size(); }

    virtual void funk4()  {
      for (const auto & val : data)
          cout << ' ' << val.second << '-' << val.first;
      cout << '\n';
    }

    bool funk5(const string t)  {
      if (data.find(t) != data.end())  {
         data.erase(t);
         return true;
      } else
         return false;
    }

    void funk6(const int n)  {
      auto it = data.begin();
      while (it != data.end())  {
        if ((*it).second > n) cout << it->first << ' ';
        it++;
      }
    }
};


class B : public A {
  private:
    string txt;

  public:
    B(const string t)  {  txt = t;  }

    int funk3()  {  return txt.length(); }

    void funk4()  {
      cout << txt << ':';
      A::funk4();
    }
};
```

```
int main() {
  A* aa = new A;
  B* bb = new B("Reidar");

  bb->funk1("Bukse", 8);   bb->funk1("Sokker", 20);   bb->funk1("Genser", 12);
  bb->funk4();

  cout << (*bb).funk2("Sokker") << ' ' << bb->funk2("Bukser") << ' '
       << aa->funk2("Genser") << '\n';

  cout << aa->funk3() << ' ' << bb->funk3() << '\n';

  bb->funk1("Jakke", 2);  bb->funk1("Sko", 6);  bb->funk1("Boxer", 12);
  bb->funk5("Bukse");     bb->funk5("Sokker");  bb->funk5("Skjorte");
  bb->funk4();

  bb->funk6(6);    cout << '\n';

  return 0;
}
```

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

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


int main()  {
  string str, str2 = "Dette er verdens fineste, enkleste og beste tekst!";
  vector <char> txt;
  list <char>  txt2;
  int n = 0;

  cout << count(str2.begin(), str2.end(), 'e') << '\n';

  for (int i = 0;  i < str2.length();  i++)
      txt.push_back(str2[i]);
  for (const auto & val : txt) cout << val;   cout << '\n';

  str = str2.substr(9, 8) + str2.substr(38, 5);  cout << str << '\n';

  auto it = txt.begin();
  while (it != txt.end()  &&  n < 12)  {
    if (*it != 'e'  &&  isalpha(*it))
       txt2.push_front(*it);
    it++;   n++;
  }
  for (const auto & val : txt2) cout << val;   cout << '\n';

  n = str2.find(str.substr(8, 5));
  cout << (n != string::npos ? n : -99) << '\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 arv, datamedlemmer og (ferdiglagde) funksjoner inni klassene, global variabel, `main` og `skrivMeny`.
Husk også på funksjonene på `LesData2.h`. Bruk *alt* dette *svært* aktivt.

Det skal holdes orden på ulike filmer som går på *en* kino (med mange saler og til ulike tidspunkter i løpet av en dag) og antall besøkende pr.forestilling av filmen (dette er *ikke* et billettbestillingsprogram).

Datastrukturen

Datastrukturen består *kun* av *listen* `gFilmer`. Hver film er av typen `Alle` (der det *ikke* er noen aldersgrense på filmen (typisk barnefilm)) eller av typen `Grense` (der den har en eller annen aldergrense (mer nedenfor)). `Alle` inneholder bl.a. en `vector` med antallet som har vært og sett filmen på de ulike forestillingene den har blitt vist.

Oppgaven

a) 12 Skriv innmaten til `void skrivAlleFilmer(), int Alle::summer()` og de to virtuelle `void skrivData()`

Den første funksjonen kommer med en egen melding om det er tomt for filmer. I motsatt fall gås det igjennom alle filmene (*vha. range-based for-løkke*). For hver enkelt film skrives dets nummer (fra 1 og oppover), og (vha. de to virtuelle funksjonene): tittel, salnummer, klokkeslett på formen TT:MM (der MM *alltid er to sifre*) og *totalantallet* (*ikke* antallet som har vært på hver enkelt forestilling av den) som har sett filmen. Dette siste sørger funksjonen `summer` for å beregne (se evt. Doxygen-kommentar i vedlegget). *Om* filmen har en aldersgrense, så skrives dette også ut. Utskriften stanser for hver femtende film, og venter på at brukeren taster ENTER.

b) 11 Skriv innmaten til `void nyFilm()` og de to virtuelle `void lesData()`

Den første funksjonen spørr og sikrer om filmen har aldersgrense eller ei. Aktuelt objekt opprettes, dets data leses inn (vha de to virtuelle funksjonene), innleste data skrives ut på skjermen (som en kvittering), og den legges inn bakerst i listen. *Alle* data, *unntatt* innholdet i `besokende` (som gjøres i neste oppgave), i de to klassene skal leses inn.
Det er aldersgrensene 6, 9, 12, 15 og 18 som er lovlige verdier. Det *skal sikres* at det *kun* er disse som leses inn og lagres. Derfor kan *ikke* bare `lesInt(…)` brukes.
NB1: Husk at klokkeslettet *skal* lagres på formen TTMM (som *en* `int`)
NB2: Du skal slippe å sjekke for duplikate titler, eller at filmer går samtidig i samme saler.

c) 8 Skriv innmaten til `void registrerBesokende()` og `void Alle::registrer()`

Den første funksjonen går igjennom alle filmene *ved bruk av iterator*. For hver film skrives dets data (*ikke* nummer) og det leses og lagres (vha. den andre funksjonen) totalt antall besøkende på filmen *den* dagen. **NB:** Kommandoen 'R' kjøres *en* gang *helt til slutt hver dag*, når man vet hvor mange som har sett *alle* filmene i *alle* salene *den* dagen.

d) 11 **Skriv innmaten til** `void fjernFilm()` **og** `void Alle::skrivTilFil()`
Den første funksjonen skriver først ut *alt* om *alle* filmene. Deretter spørres brukeren om hvilket filmnummer som skal fjernes (fordi den skal slutte å gå). Velges '0' (null) vil ingen film fjernes. Ellers vil filmens data skrives *bakerst* på filen «UTGAATTE.DTA» (vha. den andre funksjonen) og *alt* ifm. filmen slettes fra hukommelsen. Den andre funksjonen skriver *en* linje (altså *bakerst*) på filen med: tittel, totalantallet (med ledeteksten «Besøkende totalt:») som har sett filmen og antallet som har hvert på hver forestilling (med ledeteksten «Pr.forestilling:»).
Hint: Du må nok bruke iterator for å finne og slette aktuelle film.

e) 10 **Skriv innmaten til** `void mestTilstede()` **og all annen kode som trengs**
for å få finne og skrive ut hvilken film som i *gjennomsnitt* pr.forestilling (*ikke* totalt) flest har sett. Er det flere filmer med likt snitt, skrives *kun* den første ut. Har ingen filmer noe snitt, kommer det melding om dette. **NB:** Den med høyest snitt er *ikke* nødvendigvis den første i listen, dersom kommandoen 'O' er utført (neste oppgave), for der sorteres det etter *totalantallet*.

f) 8 **Skriv innmaten til** `void sorterFilmene()`
Først og sist skrives *alle* filmene ut på skjermen. Mellom dette sorteres det etter *totalantallet* som har sett filmen. Det største tallet skal ligge først (altså i avtagende verdi-rekkefølge).
Det *skal* bl.a. brukes lambda-funksjon til dette.
NB: Legg merke til at filmene på denne måten vil endre nummer, men det er helt greit, ift. det som skjer i resten av dette programmet.

g) 10 **Skriv innmaten til** `void lesFraFil()` **og**
de to constructorene med `inn` som parameter
Disse funksjonene sørger til sammen for at *hele* datastrukturen leses inn fra filen «KINO.DTA». På filen *skal* bl.a. antall filmer ligge/være. Formatet på filen bestemmer du ellers helt selv, men **det *skal* oppgis som en del av besvarelsen**.

Annet (klargjørende):

- Så lenge en film går, så foregår det i samme sal og til samme klokkeslett.
- 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)forestilling!

FrodeH

Vedlegg til PROG1003, 4.mai 2026: Halvferdig programkode

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

const int SALER  =   6;    ///<  Antall saler på kinoen.
const int START  =  11;    ///<  Klokkeslett for dagens første forestilling.
const int SLUTT  =  22;    ///<  Klokkeslett for dagens siste forestilling.
const int BESOK  = 500;    ///<  Maksimum antall besøkende på en film EN dag.

/**
 *  Baseklassen 'Alle' med dens tittel, hvilken salnummer og klokkeslett
 *  den går, samt antall besøkende på hver ulike forestilling av filmen.
 */
class Alle  {
  private:
    string tittel;          //  Filmens tittel.
    int    salNr,           //  FAST salnummer filmen går i HVER dag.
           klokkeslett;     //  FAST klokkeslett filmen vises - på formen: TTMM
    vector <int> besokende; //  Antall besøkende på hver av forestillingene.
  public:
    Alle()  { salNr = klokkeslett = 0;  }                    //  Ferdiglaget
    Alle(ifstream & inn);                                    //  Oppgave 2G
    virtual void lesData();                                  //  Oppgave 2B
    void registrer();                                        //  Oppgave 2C
    virtual void skrivData() const;                          //  Oppgave 2A
    void skrivTilFil() const;                                //  Oppgave 2D
    int summer() const;                                      //  Oppgave 2A
};


/**
 *  Avledet klasse 'Aldersgrense' med kun dets aldersgrense.
 */
class Grense : public Alle  {
  private:
    int grense;             //  Filmens aldersgrense.
  public:
    Grense()  {  grense = 0;  }                              //  Ferdiglaget
    Grense(ifstream & inn);                                  //  Oppgave 2G
    virtual void lesData();                                  //  Oppgave 2B
    virtual void skrivData() const;                          //  Oppgave 2A
};

void fjernFilm();                                            //  Oppgave 2D
void lesFraFil();                                            //  Oppgave 2G
void mestTilstede();                                         //  Oppgave 2E
void nyFilm();                                               //  Oppgave 2B
void registrerBesokende();                                   //  Oppgave 2C
void skrivAlleFilmer();                                      //  Oppgave 2A
void skrivMeny();                                            //  Ferdiglaget
void sorterFilmene();                                        //  Oppgave 2F


list <Alle*> gFilmer;                  ///<  ALLE NÅVÆRENDE aktuelle filmer.


/**
 *  Hovedprogrammet.
 */
```

```
int main() {
  char valg;
  int nr;

  lesFraFil();                                                    //  Oppgave 2G
  skrivMeny();
  valg = lesChar("\nKommando");

  while (valg != 'Q') {
     switch (valg) {
       case 'S': skrivAlleFilmer();          break;          //  Oppgave 2A
       case 'N': nyFilm();                   break;          //  Oppgave 2B
       case 'R': registrerBesokende();       break;          //  Oppgave 2C
       case 'F': fjernFilm();                break;          //  Oppgave 2D
       case 'M': mestTilstede();             break;          //  Oppgave 2E
       case 'O': sorterFilmene();            break;          //  Oppgave 2F
       default:  skrivMeny();                break;
     }
     valg = lesChar("\nKommando");
  }
  cout << "\n\n";
  return 0;
}

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

/**
 *  Oppgave 2B - Leser inn 'Alle' sine hoveddata.
 */
void Alle::lesData()  {                        /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2C - Leser og registrerer antall besokende på filmen EN gitt dag.
 */
void Alle::registrer()  {                      /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2A - Skriver ut ALT om 'Alle' på skjermen.
 */
void Alle::skrivData() const  {                /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2D - Skriver tittel og besøkende BAKERST på fil (appender).
 */
void Alle::skrivTilFil() const  {              /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2A - Summerer og returnerer summen av 'besokende's elementer.
 *  @return   Totalsummen av alle elementene i 'besokende'
 */
int Alle::summer()  const  {                   /*    LAG INNMATEN    */      }


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

```
/**
 *  Oppgave 2B - Leser inn 'Grense' sine hoveddata.
 */
void Grense::lesData()  {                       /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2A - Skriver ut ALT om 'Grense' på skjermen.
 */
void Grense::skrivData() const  {               /*    LAG INNMATEN    */      }


// ---------------------------------------------------------------------------
//                        DEFINISJON AV ANDRE FUNKSJONER:
// ---------------------------------------------------------------------------
/**
 *  Oppgave 2A - Skriver ut ALT om ALLE filmer.
 */
void skrivAlleFilmer()  {                       /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2B - Legger inn en ny film.
 */
void nyFilm()  {                                /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2C - Registrerer ALLE besøkende i ALLE saler
 *               på ALLE forestillinger på EN gitt dag.
 */
void registrerBesokende()  {                    /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2D - Fjerner (om aktuelt) en film.
 */
void fjernFilm()  {                             /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2E - Skriver filmen det gjennomsnittlig har vært flest tilstede på.
 */
void mestTilstede()  {                          /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2F - Sorterer filmene etter totalantallet som har sett filmen.
*/
void sorterFilmene()  {                         /*    LAG INNMATEN    */      }

/**
 *  Oppgave 2G - Leser ALLE kinoens nåværende filmer inn fra fil.
 */
void lesFraFil()  {                             /*    LAG INNMATEN    */      }


/**
 *  Skriver programmets menyvalg/muligheter på skjermen.
 */
void skrivMeny()  {
  cout << "\nFolgende kommandoer er tilgjengelige:\n"
       << "   S  - Skriv alle filmer\n"
       << "   N  - Ny film\n"
       << "   R  - Registrer besokende\n"
       << "   F  - Fjern/slett en film\n"
       << "   M  - filmen det har vaert gjennomsnittlig Mest/flest paa\n"
       << "   O  - sOrter filmene ut fra gjennomsnittlig mest/flest\n"
       << "   Q  - Quit / avslutt\n";
}
```