NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Kontinuasjonseksamensoppgave i PROG1003 – Objekt-orientert programmering

Faglig kontakt under eksamen:	**Frode Haug**
Eksamensdato:	**7.august 2025**
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>
using namespace std;


class A  {
  private:
    int nr;
    string navn;

  public:
    A(int nr, string navn)   {  this->nr = nr;  this->navn = navn;  }
    virtual void funk1()     {  cout << nr << ": " << navn << ' ';  }
    bool funk2(char c1, char c2)
                             {  return (navn.find(c1) != string::npos  ||
                                        navn.find(c2, 3) != string::npos);  }
    void funk3(string nvn1, string nvn2)
                             {  navn += (' ' + nvn1);
                                navn.insert(navn.find(' '), ' ' + nvn2);  }
    virtual int funk4()      {  return (nr / navn.length());  }
    void funk5(A & a, int n) {  navn += a.navn.substr(n, a.navn.length()-n);  }
};


class B : public A  {
  private:
    int antall;
  public:
    B(int nr, string navn, int ant) : A(nr, navn)  {  antall = ant;  }
    void funk1()  {  A::funk1();  cout << "Barn: " << antall << "   "; };
    int  funk4()  {  int n = A::funk4();
                     return ((n > antall) ? antall*3 : n);  }
};


int main() {
  B b1(29, "Marit", 2), b2(22, "Inger", 0), *b3 = new B(17, "Ann Helen", 1);

  b1.funk1();    b3->funk1();          cout << '\n';
  cout << b1.funk2('r', 'A')  << ' ' << b2.funk2('r', 'A') << ' '
       << b3->funk2('r', 'A') << '\n';

  b1.funk3("Olauga", "Gunhilde");  b1.funk1();  cout << '\n';

  cout << b2.funk4() << ' ' << b3->funk4() << '\n';

  b2.funk5(*b3, 3);  b2.funk1();  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>
using namespace std;


int main()  {
  vector <char> txt1(10, '-');
  string txt2 = "Jordbaer og ansjosss", txt4;
  list <char> txt3;
  int n;

  for (const auto & val : txt1) cout << val;    cout << '\n';

  n = 0;
  for (auto val : txt1) {   val = txt2[++n];  n++;  }
  for (const auto & val : txt1) cout << val;    cout << '\n';

  n = 0;
  for (auto & val : txt1) { val = txt2[++n];  n++;  }
  for (const auto & val : txt1) cout << val;    cout << '\n';

  n = 0;
  auto it = txt2.begin();
  while (*it != ' ')  {
    if (n % 2 == 1)  txt3.push_back(*it);
    else             txt3.push_front(txt2[n]);
    n+=2;   it++;
  }
  for (const auto & val : txt3) cout << val;    cout << '\n';

  for (const auto & val : txt3)  txt4 += val;
  txt4.erase(txt4.length()-1);
  cout << txt2.find_first_of(txt4) << '\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, de to klassene med arv, datamedlemmer og (*ferdiglagde*) funksjoner både inni (to stk constructorer) klassene og to stk utenfor, global variabel og `main`. Husk også på funksjonene på `LesData2.h`. Bruk *alt* dette *svært* aktivt.

Det skal holdes orden på ulike arrangement uten eller med påmelding.
(Et arrangement kan være absolutt alt fra et julebord til et orienteringsløp.)

Datastrukturen

Datastrukturen består *kun* av *mapen* `gArrangementene`. Denne har arrangementsnummeret som key, og en peker til objekter av de to ulike klassene (`Arrangement` og `Paamelding`).
Legg merke til alle kommentarene i vedlegget ifm. begge klassene.

Oppgaven

a)

Skriv innmaten til

```
void nyttArrangement()
void Arrangement::lesData()
```

Den første funksjonen spør først brukeren *og sikrer* om arrangemmentet er uten påmelding ('A') eller med ('P'). Aktuelt objekt opprettes. Alle dets data (som er i `Arrangement`) leses inn vha. den andre *ikke-virtuelle* funksjonen. Det nye objektet legges til slutt inn i datastrukturen. Et nytt arrangement får automatisk et nummer, som er 1-en høyere/mer enn den hittil bakerste/siste i map'en. Til dette brukes den ferdiglagde funksjonen `hentSisteBrukteNr()`.
NB: Navnet til påmeldte skal altså *ikke* leses inn her, det gjøres i oppgave 2d.

b)

Skriv innmaten til

```
void skrivAlleArrangementer()
```

og de to virtuelle funksjonene

```
void skrivData()
```

Den første funksjonen kommer med en egen melding om ingen arrangementer finnes.
I motsatt fall går den igjennom alle arrangementene. For hvert skrives kun dets nummer, og vha. de virtuelle funksjonene: kun navn og dato (på formen: DD/MM-20ÅÅ Husk hvilken form dette er lagret på!). For `Paamelding` skrives i tillegg teksten «PAAMELDING».

c)

Skriv innmaten til

```
void skrivEttArrangement()
```

og de to virtuelle funksjonene

```
void skrivAlt()
```

Den første funksjonen leser først et nummer mellom 1-en og det hittil siste brukte. Finnes ingen arrangement med dette nummeret, kommer det en egen melding. I motsatt fall kalles den andre funksjonen for vedkommende objekt. `Arrangement::skrivAlt` skriver ut *eksakt* det samme som klassens `skrivData`. I tillegg skriver den ut de tre andre datamedlemmene i klassen. `Paamelding::skrivAlt` sørger for at *alt* det arvede fra `Arrangement` skrives ut, pluss at den skriver ut navnet til alle de påmeldte på arrangementet.

d) **Skriv innmaten til** `void endreArrangement()`
og de to virtuelle funksjonene `void endre()`
Den første funksjonen gjør *eksakt* det sammen som den første i oppgave 2c, bare at aktuell `endre`-funksjon kalles i stedet. `Arrangement::endre` tilbyr brukeren ('J/N') *kun* å endre på `klokkeslett` og/eller `ansvarlig`, og gjør dette om ønskelig. `Paamelding::endre` gjør *eksakt* det samme, men lar i tillegg brukeren legge inn navnene på nye påmeldte inntil kun `ENTER` tastes/skrives. *Til slutt sorteres vectoren i stigende rekkefølge.*

e) **Skriv innmaten til** `void slettArrangementer()`
Funksjonen skriver først og til slutt ut *hoved*dataene om *alle* arrangementene (jfr. oppgave 2b).
Ellers spør den om nummeret på det første og det siste arrangementet som skal slettes/fjernes.
NB: Husk at de siste nummeret må være større eller lik det første.
Finnes *ikke minst* en av disse, kommer det egne meldinger, og funksjonen avsluttes.
I motsatt fall slettes/fjernes *alle* arrangementer som har numre mellom de to gitte verdiene (*inkludert* de to numrene).

f) **Skriv innmaten til** `void deltagersArrangementer()`
og all annen kode som trengs for å få skrevet ut nummeret på *alle* arrangementer en navngitt person deltar på. Store og små bokstaver skal telle likt ('æøå' kan du ignorere).

g) **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 «ARRANGEMENT.DTA». På filen *skal* bl.a. antall arrangementer ligge/være.
Formatet på filen bestemmer du ellers helt selv, men **det *skal* oppgis som en del av besvarelsen**.
Vectorene med ethvert arrangements påmeldte skal etterpå være sorterte i stigende rekkefølge.

Annet (klargjørende):

- Skulle det skje at de(n) bakerste/siste arrangement(ene) slettes/fjernes (jfr. oppgave 2e), så vil nye arrangementer (oppgave 2a) få numre som tidligere var i bruk. Dette er helt greit.
- 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 opplevelse av eksamensarrangementet!

FrodeH

Vedlegg til PROG1003, august 2025: Halvferdig programkode

```
#include <iostream>           //  cout, cin
#include <fstream>            //  ifstream
#include <string>
#include <vector>
#include <map>
#include <algorithm>          //  sort
#include <cctype>             //  toupper
#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.


/**
 *  Baseklassen 'Arrangement' med dets navn, sted og (kun) ansvarligs navn,
 *    samt datoen og klokkeslettet for arrangementet.
 */
class Arrangement  {
  private:
    string navn,              //  Arrangementets navn.
           sted,              //  Stedet det foregår på.
           ansvarlig;         //  KUN ansvarligs navn (IKKE tlf/mail/adresse)
    int    dato,              //  På formen:  ÅÅMMDD
           klokkeslett;       //  På formen:  TTMM
  public:
    Arrangement()  {  dato = FDATO;  klokkeslett = 0;  }      //  Ferdiglaget
    Arrangement(ifstream & inn);                              //  Oppgave 2G
    virtual void endre();                                     //  Oppgave 2D
             void lesData();                                  //  Oppgave 2A
    virtual void skrivAlt()  const;                           //  Oppgave 2C
    virtual void skrivData() const;                           //  Oppgave 2B
};


/**
 *  Avledet klasse 'Paamelding' med alle arrangementets påmeldte navn.
 */
class Paamelding : public Arrangement  {
  private:
    vector <string> paameldte;
  public:
    Paamelding()  {  }                                        //  Ferdiglaget
    Paamelding(ifstream & inn);                               //  Oppgave 2G
    virtual void endre();                                     //  Oppgave 2D
    virtual void skrivAlt()  const;                           //  Oppgave 2C
    virtual void skrivData() const;                           //  Oppgave 2B
};


void   deltagersArrangementer();                              //  Oppgave 2F
void   endreArrangement();                                    //  Oppgave 2D
int    hentSisteBrukteNr();                                   //  Ferdiglaget
void   lesFraFil();                                           //  Oppgave 2G
void   nyttArrangement();                                     //  Oppgave 2A
void   skrivAlleArrangementer();                              //  Oppgave 2B
void   skrivEttArrangement();                                 //  Oppgave 2C
void   skrivMeny();                                           //  Ferdiglaget
void   slettArrangementer();                                  //  Oppgave 2E
```

```
                                    ///<  Datastrukturen med ALLE arrangementene:
map <int, Arrangement*> gArrangementene;


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

  lesFraFil();                                                  //  Oppgave 2G

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

  while (valg != 'Q') {
     switch (valg) {
       case 'N': nyttArrangement();            break;       //  Oppgave 2A
       case 'A': skrivAlleArrangementer();     break;       //  Oppgave 2B
       case '1': skrivEttArrangement();        break;       //  Oppgave 2C
       case 'E': endreArrangement();           break;       //  Oppgave 2D
       case 'S': slettArrangementer();         break;       //  Oppgave 2E
       case 'D': deltagersArrangementer();     break;       //  Oppgave 2F
       default:  skrivMeny();                  break;
     }
     valg = lesChar("\nKommando");
  }

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


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

/**
 *  Oppgave 2G - Leser inn ALLE Arrangementets data fra fil.
 */
Arrangement::Arrangement(ifstream & inn)  {          /*    LAG INNMATEN      */    }


/**
 *  Oppgave 2D - Kan evt endre på Arrangementets dato og/eller klokkeslett.
 */
void Arrangement::endre()  {                         /*    LAG INNMATEN      */    }


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


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


/**
 *  Oppgave 2B - Skriver ut Arrangementets navn og dato (DD/MM-20ÅÅ).
 */
void Arrangement::skrivData() const  {               /*    LAG INNMATEN      */    }
```

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

/**
 *  Oppgave 2G - Leser inn ALLE Paameldings data fra fil.
 */
Paamelding::Paamelding(ifstream & inn) : Arrangement(inn)  {/*  LAG INNMATEN  */  }


/**
 *  Oppgave 2D - Kan endre på Arrangementets dato/klokkeslett og påmeldte.
 */
void Paamelding::endre()  {                          /*    LAG INNMATEN      */    }


/**
 *  Oppgave 2C - Skriver ut på skjermen ALLE Paameldtes data.
 */
void Paamelding::skrivAlt()  const  {                /*    LAG INNMATEN      */    }


/**
 *  Oppgave 2B - Skriver ut Paameldings HOVEDdata.
 */
void Paamelding::skrivData() const  {                /*    LAG INNMATEN      */    }


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

/**
 *  Oppgave 2F - Finner og skriver alle en gitt deltagers arrangementer.
 */
void deltagersArrangementer()  {                     /*    LAG INNMATEN      */    }


/**
 *  Oppgave 2D - Endrer noen av ETT arrangements data, inkl evt påmeldte.
 */
void endreArrangement()  {                           /*    LAG INNMATEN      */    }


/**
 *  Finner og returnerer det aller siste/bakerste arrangementets nummer.
 *
 *  @return   Det aller siste/bakerste arrangementets nummer
 */
int hentSisteBrukteNr()  {
  if (gArrangementene.size() == 0) return 0; // Tomt - returnerer '0'.
  auto it = gArrangementene.end();   it--;   // iterator settes til den siste.
  return (it->first);                        // Returnerer dens nummer/key.
}


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


/**
 *  Oppgave 2A - Legger inn ett nytt arrangement.
 */
void nyttArrangement()  {                            /*    LAG INNMATEN      */    }
```

```
/**
 *  Oppgave 2B - Skriver ut noen hoveddata for ALLE arrangementer.
 */
void skrivAlleArrangementer()  {                    /*    LAG INNMATEN      */    }


/**
 *  Oppgave 2C - Skriver ALLE data om ETT arrangement.
 */
void skrivEttArrangement()  {                       /*    LAG INNMATEN      */    }


/**
 *  Skriver programmets menyvalg/muligheter på skjermen.
 */
void skrivMeny()  {
  cout << "\nFolgende kommandoer er tilgjengelige:\n"
       << "   N  - Nytt arrangement\n"
       << "   A  - skriv Alle arrangementer\n"
       << "   1  - skriv ETT arrangement\n"
       << "   E  - Endre (og/eller evt. paameldte) arrangement\n"
       << "   S  - Slett ett eller flere arrangementer\n"
       << "   D  - alle arrangementer en Deltager er paa\n"
       << "   Q  - Quit / avslutt\n";
}


/**
 *  Oppgave 2E - Sletter ett eller flere arrangementer i et nummerintervall.
 */
void slettArrangementer()  {                        /*    LAG INNMATEN      */    }
```