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:	**7.august 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>
#include <vector>
using namespace std;


const int MAX = 4;


class A  {
  protected:
    vector <int> tall;

  public:
    A()  {  }
    A(const int n) {  for (int i = 1;  i < MAX;  i++)  tall.push_back(i*n); }
    void display() const { for (const auto & val : tall) cout << ' ' << val;}
    virtual bool funk(const int t) const  {  return (tall.at(1) != 15);  }
};


class B : public A {
  private:
    string txt;

  public:
    B(const int n, const string t)
      {  txt = t;  for (int i = MAX;  i < MAX*2;  i++) tall.push_back(i*n); }
    void display2()
      {  cout << tall.front() << ' ' << tall.back() << ' ' << tall.size();  }
    virtual bool funk(const string t) const  {  return (!(txt == t));  }
};


int main() {
  A* obj1 = new A(5);
  B* obj2 = new B(3, "Raches");
  A* obj3 = new B(4, "Therma");

  obj1->display();   cout << '\n';
  obj2->display();   cout << '\n';
  cout << ((obj2->funk("Raches")) ? "Nas" : "Evdilos") << '\n';
  cout << ((obj3->funk(15)) ? "Nas" : "Evdilos")       << '\n';
  obj2->display2();  cout << ' ';
                     ((dynamic_cast <B*> (obj3)))->display2(); 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>
using namespace std;


int main() {
   vector <string> tekst1 { "Eple", "Banan", "Appelsin", "Sitron", "Mango" };
   vector <string> tekst2;

   tekst2.push_back("Gulrot");  tekst2.push_back("Brokkoli");
   tekst2.push_back("Salat");   tekst2.push_back("Potet");
   sort(tekst2.begin(), tekst2.end());

   for (const auto & val : tekst2) cout << val << "  ";         cout << '\n';

   auto it = tekst1.begin();  it++;  it++;
   auto it2 = it;
   for (  ;  it != tekst1.end();  it++)  cout << *it << "  ";   cout << '\n';

   it2++;
   tekst1.erase(it2, tekst1.end());
   tekst1.pop_back();    tekst1.push_back("Kiwi");
   for (const auto & val : tekst1) cout << val << "  ";         cout << '\n';


   list <char> tegn { 'A', 'R', 'S', 'E', 'N', 'A', 'L' };
   tegn.sort();   tegn.reverse();
   for (auto it3 = tegn.rbegin();  it3 != tegn.rend();  it3++)
       cout << *it3 << ' ';                                    cout << '\n';

   tegn.reverse();  tegn.remove('A');
   auto it4 = tegn.begin();  it4++;  it4++;
   tegn.erase(it4);
   cout << tegn.front() << ' ' << tegn.back() << ' ' << tegn.size() << ' ';
   it4 = tegn.begin();
   while (it4 != tegn.end())  cout << *it4++ << ' ' ;          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 `constene`, `enumen`, klassene med datamedlemmer og (*ferdiglagde*) funksjoner (inni/utenfor klassene), global variabel, `main` og `skrivMeny()`. Husk også på de ferdiglagde funksjonene på `LesData2.h`. Bruk *alt* dette *svært* aktivt.

Vi skal lage et program som holder orden på ulikt militært personell sin tagning av ulike (ferdighets)merker.

Datastrukturen

Datastrukturen består kun av *mapen* `gPersonellet`. Denne har som `first` vedkommendes etternavn. Dette er unikt, dvs. ingen skal faktisk ha samme etternavn(!).

Oppgaven

a) 7 Skriv innmaten til `void nyttPersonell()` og `void Personell::lesData()`

Den første funksjonen leser inn et *nytt unikt* etternavn. Finnes denne fra før, kommer det en egen melding. I motsatt fall opprettes et nytt `Personell`, alle dets tre `string`-data leses inn (vha. den andre funksjonen), og til slutt legges den inn i datastrukturen.

b) 6 Skriv innmaten til `void skrivAltPersonell()` og `void Personell::skrivHovedData()`

Den første funksjonen går igjennom *alt* personellet. For hver skrives (bl.a. vha. den andre funksjonen) dens etternavn, fornavn, grad, (tjeneste)sted og *antall* merker (men *ikke detaljene* om hvert merke, det gjøres i oppgave 2f).

c) 6 Skriv innmaten til

```
Merke* Personell::lagMerke(const char type, const string t, const Valor v)
```

`type` er 'P' (for `PoengMerke`) eller 'T' (for `TidMerke`). Ut ifra denne bokstaven lages og returneres en peker til et aktuelt nytt merke. Til dette brukes også de to andre parametrene ovenfor. Er `type` ugyldig kommer det en melding, og `nullptr` returneres.

d) 4 Skriv innmaten til `void nyttMerke()`

Først spørres det om et etternavn. Finnes det ikke noe personell med dette etternavnet, kommer det en egen melding. Ellers tilkalles det aktuelle personelletts `nyttMerke`-funksjon (jfr. oppgave 2e).

e) 12 Skriv innmaten til `void Personell::nyttMerke()` og de tre virtuelle `lesData`-funksjonene i de tre `Merke`-klassene.

Den første funksjonen skriver først ut *alt* personellets data. Deretter leses et merkes tittel, dets valør (bruk ferdiglaget funksjon) og om det er et P(oeng)- eller et T(ids)-merke. Nytt aktuelt merke opprettes (vha. funksjonen fra 2c). *Alle* dets resterende data leses så inn (vha. de tre virtuelle funksjonene. Husk at tiden skal lagres i *totalt antall sekunder*.), før det legges inn *bakerst*/til slutt i vectoren (skal altså *ikke* sorteres på noe kriterie). Til sist skrives dataene om *alle* merkene til vedkommende personell ut på skjermen.

f) **14** **Skriv innmaten til** `void skrivAltOmEttPersonell()`, `void Personell::skrivAlleData()` **og de tre virtuelle `skrivData`-funksjonene i de tre `Merke`-klassene.**

Den første funksjonen kommer med en egen melding om ingen personell finnes. Ellers spørres det om et etternavn. Egen melding om ingen slik finnes. I motsatt fall skrives *alle* data om vedkommende ut vha. den andre funksjonen. Denne funksjonen skriver først det aktuelle personellets hoveddata, før det til slutt skriver *alle* data om *alle* merkene. Til dette siste brukes de tre virtuelle funksjonene. Det *skal* skrives om merket er et «POENG»- eller «TID»-merke. `valor` *skal* skrives som tekstene «Bronse/Sølv/Gull». Dato *skal* skrives på formen: DD/MM-ÅÅ og tiden skal skrives på formen: time : min : sek (selv om dette altså er lagret som *totalt* antall sekunder).

g) **5** **Skriv innmaten til**

```
int Personell::antallMerker(const string tittel, const Valor valor)
```

Funksjonen finner ut og returnerer totalt antall merker vedkommende personell har av aktuelle tittel og valør. Bruk bl.a. ferdiglagde funksjoner til dette.

h) **8** **Skriv innmaten til** `void Personell::nyttMerke()` **- versjon 2**

Det skal her lages en ny versjon av den første funksjonen i oppgave 2e. Den er helt identisk til det angitt tidligere, bortsett fra et det *ikke* spørres om merkets valør, da dette skal automatisk bestemmes/settes. Til dette må bl.a. funksjonene i 2g og 2c brukes. Reglene for merketagelse er som følger: Når man har tre bronse av et merke, får man den fjerde gangen sølv av merket. Når man har tre sølv, får man gull den fjerde gangen. Deretter kan man få så mange gull man ønsker av det samme merket. Altså først den syvende gangen oppnår man gull av det samme merket.
Lite hint: Sikkert lurt å starte med å sjekke antallet av Gull, deretter antall Sølv, så Bronse.

i) **10** **Skriv innmaten til** `void lesFraFil()` **og *alle* de fire constructorene (som tar et filobjekt som parameter) i *alle* klassene**

Disse funksjonene sørger til sammen for at *hele* mapen med personell leses inn fra filen «MERKER.DTA». Formatet på filen bestemmer du helt selv, men **dette *skal* oppgis som en del av besvarelsen**.

Annet (klargjørende):

- Merketittelen lagres for hver gang inni hvert merke. Dette kunne selvsagt ha vært gjort mer effektivt (men er altså løst slik her). Vi forutsetter dessuten at brukeren skrive denne likt og korrekt hver gang.
- Det er ikke mulig å endre et personell sin grad eller (tjeneste)sted (som selvsagt er mulig i praksis).
- Brukeren må stadig oppgi merketype (P/T), selv dette egentlig er kjent når tittelen er kjent.
- Det er ikke laget noen destructor eller annen kode for å slette/delete det er sagt «new» om.
- 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 med tagningen av «Eksamen bestått» - merket!

FrodeH

Vedlegg til PROG1003, august 2023: Halvferdig programkode

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


const float MINPOENG =     1,      ///<  Minimum antall poeng å få.
            MAXPOENG = 10000;      ///<  Maksimum antall poeng å få.
const int   MINTIME  =     0,      ///<  Minimum time brukt.
            MAXTIME  =    10;      ///<  Maksimum time brukt.

/**
 *  Valor (merkenes ulike valører: Bronse, Sølv eller Gull).
 */
enum Valor {  Bronse,  Solv,  Gull  };


class Merke;                      //  PRE-deklarasjon, da 'Personell' henviser
                                  //    til 'Merke' som defineres lengre nede.
/**
 *  Personell (med fornavn, miltær grad og tjenestested).
 */
class Personell {
  private:                         //  'etternavn' er 'first' i map'en.
    string fornavn,
           grad,                   //  Militær grad.
           sted;                   //  Tjenestested.
    vector <Merke*> merkene;       //  Alle merkene totalt tatt.

   int antallMerker(const string tittel, const Valor valor) const;  // Oppg.2G
   Merke* lagMerke(const char type, const string t, const Valor v); // Oppg.2C
   Valor  lesValor() const;                    //  (Ferdiglaget nedenfor)

  public:
    Personell()  {    }
    Personell(ifstream & inn);                 //  Oppgave 2I
    void lesData();                            //  Oppgave 2A
    void nyttMerke();                          //  Oppgave 2E  og  2H
    void skrivHovedData() const;               //  Oppgave 2B
    void skrivAlleData() const;                //  Oppgave 2F
};


/**
 *  Merke (med tittel/beskrivelse, dato tatt og merkets valør).
 */
class Merke {
  private:
    string tittel;                 //  Tittel/beskrivelse av merket.
    int    dato;                   //  På formen:  ÅÅMMDD   (år-måned-dag).
    Valor  valor;                  //  Merkets valør:  Bronse, Solv eller Gull.

  public:
    Merke(const string t, const Valor v)  {  tittel = t;  valor = v; }
    Merke(ifstream & inn);                     //  Oppgave 2I
    string hentTittel()  {  return tittel;  }
    Valor  hentValor()   {  return valor;   }
    virtual void lesData();                    //  Oppgave 2E
    virtual void skrivData() const;            //  Oppgave 2F
};
```

```
/**
 *  PoengMerke (med poengene oppnådd ifm. merketagelsen).
 */
class PoengMerke : public Merke {
  private:
    float  poeng;

  public:
    PoengMerke(const string t, const Valor v) : Merke(t, v)  {  }
    PoengMerke(ifstream & inn);               //  Oppgave 2I
    virtual void lesData();                   //  Oppgave 2E
    virtual void skrivData() const;           //  Oppgave 2F
};


/**
 *  TidMerke (tiden brukt ifm. merketagelsen).
 */
class TidMerke : public Merke {
  private:
    int tid;             //  Tid i hele sekunder (= time*3600 + min*60 + sek).

  public:
    TidMerke(const string t, const Valor v) : Merke(t, v)  {  }
    TidMerke(ifstream & inn);                 //  Oppgave 2I
    virtual void lesData();                   //  Oppgave 2E
    virtual void skrivData() const;           //  Oppgave 2F
};


void  lesFraFil();                             //  Oppgave 2I
void  nyttMerke();                             //  Oppgave 2D
void  nyttPersonell();                         //  Oppgave 2A
void  skrivAltOmEttPersonell();                //  Oppgave 2F
void  skrivAltPersonell();                     //  Oppgave 2B
void  skrivMeny();


map <string, Personell*> gPersonellet;     ///<  ALT personell som tar merker.


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

  lesFraFil();                                          //  Oppgave 2I

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

  while (valg != 'Q') {
     switch (valg) {
       case 'P':  nyttPersonell();            break;  //  Oppgave 2A
       case 'A':  skrivAltPersonell();        break;  //  Oppgave 2B
       case 'M':  nyttMerke();                break;  //  Oppgave 2D
       case 'E':  skrivAltOmEttPersonell();   break;  //  Oppgave 2F
       default:   skrivMeny();                break;
     }
     valg = lesChar("\nKommando");
  }

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

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

/**  Oppgave 2G - Finner og returnerer antallet av et gitt merke.
 *   @param    tittel  -  Merkets tittel/navn/beskrivelse
 *   @param    valor   -  Merkets valør (Bronse, Solv, Gull)
 *   @return   Antall merker med EKSAKT 'tittel' OG 'valor'
 */
int Personell::antallMerker(const string tittel, const Valor valor) const {
                                                           /*   LAG INNMATEN   */  }
/**  Oppgave 2C - Oppretter og returnerer et aktuelt merke.
 *   @param    type  -  Aktuell merketype ('P/T') å lage nytt merke av
 *   @param    t     -  Merkets tittel/navn/beskrivelse
 *   @param    v     -  Merkets valør (Bronse, Solv, Gull)
 *   @return   Peker til nyopprettet aktuelt merke
 */
Merke* Personell::lagMerke(const char type, const string t, const Valor v) {
                                                           /*   LAG INNMATEN   */  }
/**  Oppgave 2A - Leser inn ALLE egne data fra tastaturet.
 */
void Personell::lesData()  {                               /*   LAG INNMATEN   */  }

/**  Leser inn og returnerer valør for ett merke.
 *   @return   Merkevalør (Bronse, Solv, Gull)
 */
Valor Personell::lesValor() const  {
  char tegn;

  do  {
     tegn = lesChar("\tValor (B(ronse), S(olv), G(ull))"); //  Leser og sikrer
  } while (tegn != 'B'  &&  tegn != 'S'  &&  tegn != 'G'); //    lovlig tegn.

  switch (tegn)  {
    case 'B':   return Bronse;                //  Returnerer aktuell merke-
    case 'S':   return Solv;                  //    valør ut fra tegnet.
    case 'G':   return Gull;
  }
}

/**  Oppgave 2E -  Legger inn ett nytt merke (versjon 1).
 */
void Personell::nyttMerke()  {                             /*   LAG INNMATEN   */  }

/**  Oppgave 2H - Legger inn ett nytt merke (versjon 2).
 */
void Personell::nyttMerke2()  {                            /*   LAG INNMATEN   */  }

/**  Oppgave 2B - Skriver alle egne HOVEDdata ut på skjermen.
 */
void Personell::skrivHovedData() const  {                  /*   LAG INNMATEN   */  }

/**  Oppgave 2F - Skriver ALLE egne data ut på skjermen.
 */
void Personell::skrivAlleData() const  {                   /*   LAG INNMATEN   */  }


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

/**  Oppgave 2E - Leser inn ALLE egne data fra tastaturet.
 */
void Merke::lesData()  {                                   /*   LAG INNMATEN   */  }
```

```
/**  Oppgave 2F - Skriver ALLE egne data ut på skjermen.
 */
void Merke::skrivData() const  {                              /*   LAG INNMATEN   */ }


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

/**  Oppgave 2E - Leser inn ALLE egne data fra tastaturet.
 */
void PoengMerke::lesData()  {                                 /*   LAG INNMATEN   */ }

/**  Oppgave 2F - Skriver ALLE egne (og baseklassens) data ut på skjermen.
 */
void PoengMerke::skrivData() const  {                         /*   LAG INNMATEN   */ }


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

/**  Oppgave 2E - Leser inn ALLE egne data fra tastaturet.
 */
void TidMerke::lesData()  {                                   /*   LAG INNMATEN   */ }

/**  Oppgave 2F - Skriver ALLE egne (og baseklassens) data ut på skjermen.
 */
void TidMerke::skrivData() const  {                           /*   LAG INNMATEN   */ }


// ---------------------------------------------------------------------------
/** Oppgave 2I - Leser ALT personell og ALLE deres merker inn fra fil.
 */
void lesFraFil()  {                                           /*   LAG INNMATEN   */ }

/** Oppgave 2D - Legger inn (om mulig) et nytt merke.
 */
void nyttMerke()  {                                           /*   LAG INNMATEN   */ }

/** Oppgave 2A - Legger inn (om mulig) et nytt personell.
 */
void nyttPersonell()  {                                       /*   LAG INNMATEN   */ }

/** Oppgave 2F - Skriver ut ALT om ETT personell.
 */
void skrivAltOmEttPersonell()  {                              /*   LAG INNMATEN   */ }

/** Oppgave 2B - Skriver ut HOVEDdata om ALT personell.
 */
void skrivAltPersonell()  {                                   /*   LAG INNMATEN   */ }

/** Skriver programmets menyvalg/muligheter på skjermen.
 */
void skrivMeny()  {
  cout << "\nFølgende kommandoer er tilgjengelige:\n"
       << "   P  - nytt Personell\n"
       << "   A  - skriv ut Alt personellet\n"
       << "   M  - nytt Merke\n"
       << "   E  - skriv ut alt om Ett personell\n"
       << "   S  - statistikk over medaljer\n"
       << "   Q  - Quit / avslutt\n";
}
```