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:	**13.mai 2024**
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):

```
//  NB: string::npos    betyr "ikke funnet", det samme som en iterators end()


#include <iostream>
#include <string>
using namespace std;


class A  {
  protected:
     string navn, adresse;
  public:
     A()  {  }
     A(const string n, const string a)  {  navn = n;  adresse = a;  }
     virtual void display() const = 0;
     void funk1(const char c)  {    int n;
       while ((n = adresse.find(c, 0)) != string::npos) adresse[n] = 't'; }
     bool funk2()  {  return (navn.substr(1, 5) == adresse.substr(1, 5));  }
};


class B : public A {
  private:
    float hoyde;
  public:
    B(const string n, const string a, const float h) : A(n, a) { hoyde = h; }
    B(const B & b)       { navn = (b.navn + " Idar");
                           adresse = b.adresse + 'a';  hoyde = b.hoyde;   }
    void display() const { cout << navn << ", " << adresse
                                << " hoyde: " << hoyde << "    "; }
    float funk3(const B b, const float hoyde)
                         {  this->hoyde = (b.hoyde > hoyde) ? hoyde : b.hoyde;
                            return (this->hoyde); }
};


int main() {
  B  nr1("Are", "Aremark", 179.5),
     nr2("Odd", "Odda", 183.5);
  B* nr3 = new B("Morten", "Horten", 184.0);
  nr2.display();    nr1.display();            cout << '\n';
  nr2.funk1('d');   nr2.display();            cout << '\n';
  cout << (!nr3->funk2() ? "Vel" : "Joda")  <<    " ...\n";
  cout << nr1.funk3(*nr3, 183.5)    << '\n';
  B nr4(nr1);  nr4.display();  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 <string> txt {"GG", "SS", "KK", "BB", "FF", "TT", "AA", "PP", "EE"};
  list <string> txt2;
  int n;

  for (int i = 1;  i < txt.size();  i+=2)
      cout << txt[i] << ' ';   cout << '\n';

  txt2.push_back(txt[4]);
  for (int i = 4;  i > 0;  i--)  {
      txt2.push_back(txt[4-i]);
      txt2.push_front(txt[4+i]);
  }

  for (const auto & val : txt2)
      cout << val << ' ';      cout << '\n';

  txt2.sort();  txt2.reverse();

  auto it  = txt2.begin();
  auto it2 = txt2.rbegin();
  while (*it != *it2) {
    it++;  it2++;
  }
  cout << *it << ' ' << *it2 << '\n';

  while (txt2.front() != *it)  {
    cout << txt2.front() << ' ';
    txt2.pop_front();
  }
  cout << '\n';

  n = txt2.size();
  for (int i = 1;  i <= n;  i++)
      txt2.push_back(*it++);
  for (const auto & val : txt2)
      cout << val << ' ';      cout << '\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, klassen med datamedlemmer og (*ferdiglagde*) funksjoner inni klassen, global variabel, `main` og `skrivMeny`.
Husk også på funksjonene på `LesData2.h`. Bruk *alt* dette *svært* aktivt.

Det skal holdes orden på hvilke fotballbaner er person har besøkt i hvilke byer/steder og på hvilke ulike datoer vedkommende har besøkt disse.

Datastrukturen

Datastrukturen består *kun* av *mapen* `gBanene`. Alle banene har alle et *unikt* navn, og som er keyen i mapen. Hver bane inneholder bl.a. en `vector` med de ulike datoene (på formen ÅÅMMDD) som banen er besøkt.

Oppgaven

a) 12 Skriv innmaten til `void skrivAlleBaner()` `void Bane::skrivData()`

Det kommer en egen melding om ingen baner finnes. Ellers skrives antall baner ut på skjermen, samt *alle* data om hver av banene. Dette siste gjøres bl.a. vha. den andre funksjonen. Det *skal* være to linjer pr.bane, med følgende form: **Emirates Stadium, 60704 Arsenal, London**
22/4-21 17/8-22 29/9-23
Legg merke til formatet på datoen, der altså den lagrede ÅÅMMDD er gjort om til DD/MM-ÅÅ
Er det ingen besøksdatoer ennå registrert på vedkommende bane, kommer det en egen melding.

b) 8 Skriv innmaten til `void nyBane()` `void Bane::lesData()`

Den første funksjonen spør først om den nye banens unike navn. Finnes denne allerede, kommer en egen meding. I motsatt fall opprettes en ny, alle dens data (unntatt besøksdatoer) leses inn vha. den andre funksjonen, og den legges til slutt inn i datastrukturen.

c) 10 Skriv innmaten til `void nyttBesok()` `void Bane::nyttBesok()`

Den første funksjonen spør først om banens navn. Finnes denne *ikke*, kommer en egen meding. I motsatt fall registreres *ett* nytt besøk på banen. Dette siste gjøres vha. den andre funksjonen. Som leser inn en dag, måned og år (2000-2030). Før dette lagres, sørges det for at det gjøres om til noe på formen ÅÅMMDD (der de to første sifrene i årstallet altså er kuttet vekk). Vectoren *skal* holdes sortert på stigende dato.

d) **10** **Skriv innmaten til** `void skrivGitteUtvalgte()`

Det leses *og sikres* om brukeren vil se en by ('B') eller baners kapasitet ('K').
Har vedkommende valgt 'B', så spørres det om et bynavn. Deretter skrives ut *kun* navnet på *alle* baner som er i denne byen. Velges det 'K', spørres det om en minimum kapasitet for en bane. Deretter skrives navnet på alle baner, *og kun* deres kapasitet som har minst dette som kapasitet. Er det ingen baner som gir match/utskrift i de to tilfellene, kommer det en egen melding.

e) **12** **Skriv innmaten til** `void skrivGittAar()` **og all annen kode som trengs**

for å få skrevet ut navnet på alle banene som er besøkt et visst år.
Året *skal* av brukeren skrives inn som et tall i intervallet 2000-2030.
Det *skal* også brukes iterator for å gå igjennom alle banene.

f) **8** **Skriv innmaten til** `void frigiAllokertMinne()`

Sletter alt det som det er sagt `new` om mens programmet har kjørt.
Det *skal* bl.a. brukes `for_each(…)` og lambda-funksjon for å utføre/gjøre dette.

g) **10** **Skriv innmaten til**

```
void lesFraFil()
void Bane::lesFraFil(ifstream & inn)
```

Disse funksjonene sørger til sammen for at *hele* datastrukturen leses inn fra filen «BANER.DTA». På filen skal antall baner *ikke* ligge/være. Formatet på filen bestemmer du helt selv, men **det *skal* oppgis som en del av besvarelsen**.

Annet (klargjørende):

- På hver bane spiller *kun ett* lag. Men et lag må gjerne ha flere baner (det kan jo hende at den forrige er nedlagt/borte, som nylig for f.eks: Arsenal, Tottenham og West Ham).
- «By» trenger ikke nødvendigvis reelt å være en by. Det kan også være et sted, som ikke har bystatus.
- 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 groundhopping!

FrodeH

Vedlegg til PROG1003, 13.mai 2024: Halvferdig programkode

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



const int MINKAPASITET =   1000;   ///<  Minimum kapasitet på en bane.
const int MAXKAPASITET = 120000;   ///<  Maksimum kapasitet på en bane.
const int MINAAR       =   2000;   ///<  Første år for registrering av besøk.
const int MAXAAR       =   2030;   ///<  Siste år for registrering av besøk.



/**
 *  Bane med hjemmelagets navn, hvilken by/sted den ligger, kapasitet
 *  og alle datoer som man har besøkt banen.
 */
class Bane  {
  private:
    string lag, by;
    vector <int> datoer;                    //  Hver dato er på formen:  ÅÅMMDD
    int kapasitet;
  public:
    Bane()  {  kapasitet = 0;  }                        //        (ferdiglaget)
    string  hentBy() const         {  return by;  }  //        (ferdiglaget)
    int     hentKapasitet() const  {  return kapasitet;  } //  (ferdiglaget)
    void    lesData();                                  //  Oppgave 2B
    void    lesFraFil(ifstream & inn);                  //  Oppgave 2G
    void    nyttBesok();                                //  Oppgave 2C
    void    skrivData() const;                          //  Oppgave 2A
};



void frigiAllokertMinne();                              //  Oppgave 2F
void lesFraFil();                                       //  Oppgave 2G
void nyBane();                                          //  Oppgave 2B
void nyttBesok();                                       //  Oppgave 2C
void skrivAlleBaner();                                  //  Oppgave 2A
void skrivGittAar();                                    //  Oppgave 2E
void skrivGitteUtvalgte();                              //  Oppgave 2D
void skrivMeny();                                       //     (ferdiglaget)



map <string, Bane*> gBanene;       ///<  Datastrukturen med ALLE banene.
```

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

  lesFraFil();                                          //  Oppgave 2G

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

  while (valg != 'Q') {
     switch (valg) {
       case 'S': skrivAlleBaner();       break;       //  Oppgave 2A
       case 'N': nyBane();               break;       //  Oppgave 2B
       case 'B': nyttBesok();            break;       //  Oppgave 2C
       case 'U': skrivGitteUtvalgte();   break;       //  Oppgave 2D
       case 'A': skrivGittAar();         break;       //  Oppgave 2E
       default:  skrivMeny();            break;
     }
     valg = lesChar("\nKommando");
  }

  frigiAllokertMinne();                                 //  Oppgave 2F

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



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

/**
 *  Oppgave 2B - Leser inn alle data (unntatt besøk) om en bane.
 */
void Bane::lesData()  {                         /*    LAG INNMATEN    */      }



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



/**
 *  Oppgave 2C - Legger inn eN ny dato for besøk på banen.
 */
void Bane::nyttBesok()  {                       /*    LAG INNMATEN    */      }



/**
 *  Oppgave 2A - Skriver ut på skjermen ALLE data om banen.
 */
void Bane::skrivData() const  {                 /*    LAG INNMATEN    */      }
```

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

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



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



/**
 *  Oppgave 2B - Legger inn (om mulig) en ny bane.
*/
void nyBane()  {                                /*    LAG INNMATEN    */      }



/**
 *  Oppgave 2C - Legger inn (om mulig) ett nytt besøk på en bane.
*/
void nyttBesok()  {                             /*    LAG INNMATEN    */      }



/**
 *  Oppgave 2A - Skriver ut på skjermen ALLE datene om ALLE banene.
*/
void skrivAlleBaner()  {                        /*    LAG INNMATEN    */      }



/**
 *  Oppgave 2E - Skriver ut ALLE baner besøkt et gitt år.
 */
void skrivGittAar()  {                          /*    LAG INNMATEN    */      }



/**
 *  Oppgave 2D - Skriver ut ALLE baner i en gitt by eller med en viss kapasitet.
*/
void skrivGitteUtvalgte()  {                    /*    LAG INNMATEN    */      }



/**
 *  Skriver programmets menyvalg/muligheter på skjermen.
 */
void skrivMeny()  {
  cout << "\nFolgende kommandoer er tilgjengelige:\n"
       << "   S  - Skriv alle baner\n"
       << "   N  - Ny bane\n"
       << "   B  - nytt besok paa en bane\n"
       << "   U  - skriv gitte Utvalgte byer eller kapasiteter\n"
       << "   A  - skriv alle banebesok ett gitt Aar\n"
       << "   Q  - Quit / avslutt\n";
}
```