Høgskolen i Gjøvik
Avdeling for informatikk og medieteknikk

Kontinuasjonseksamen

EMNENAVN: **Objekt-orientert programmering**

EMNENUMMER: **IMT1082**

EKSAMENSDATO: **5. januar 2012**

KLASSE(R): **10HBIND* / 10HBPUA / 10HBDRA / 10HBISA / 10HBSPA**

TID: **09.00-13.00**

EMNEANSVARLIG: **Frode Haug**

ANTALL SIDER UTLEVERT: **9 (inkludert denne forside)**

TILLATTE HJELPEMIDLER: **Alle trykte og skrevne.**

- Kontroller at alle oppgavearkene er til stede.
- Innføring med penn, eventuelt trykkblyant som gir gjennomslag. Pass på så du ikke skriver på mer enn ett innføringsark om gangen (da det blir uleselige gjennomslag når flere ark ligger oppå hverandre).
- Ved innlevering skilles hvit og gul besvarelse, som legges i hvert sitt omslag.
- Oppgavetekst, kladd og blå kopi beholder kandidaten til klagefristen er over.

- Ikke skriv noe av din besvarelse på oppgavearkene. Men, i oppgavetekst der du skal fylle ut svar i tegning/tabell/kurve, skal selvsagt dette innleveres sammen med hvit besvarelse.

- Husk kandidatnummer på alle ark. Ta vare på dette nummeret til sensuren faller.

NB1: Dette oppgåvesettet inneheld minst 25% nynorske ord.

NB2: Oppgave 1a, 1b, 1c og 2 er totalt uavhengige og kan derfor løses separat.

Oppgave 1 (30%)

a) Det nedenfor stående programmet var det ønskelig at gav følgende utskrift:

3: Arsenal
1: Emirates 1 1
2: Stadium 1.8 1
0 1 1

Men, i koden er det **fem feil** av typen **syntaktiske** (som kompilatoren reagerer på) og/eller **semantiske** (logiske, som gjør at programmet ikke fungerer som ønsket). **Hvilke ?**

NB: Tallene helt til venstre på hver linje er *ikke* en del av koden, men kun ment som linjenummer, slik at du enklere kan henvise til hvor feilene er å finne.

```
 1    #include <iostream>
 2    #include <cstring>
 3    using namespace std;

 4    class Klasse1  {
 5      private:
 6          char txt[];
 7          int  nr;
 8      public:
 9          Klasse1(char* t, int n)    {  strcpy(t, txt);  nr = n;  }
10          bool lik(char* t)  {  return(!strcmp(t, txt));  }
11          virtual void display()  {  cout << nr << ":  " << txt; }
12    };

13    class Klasse2 : public Klasse1  {
14      private:
15          float tall;
16          bool  sant;
17      public:
18          Klasse2(char* t, int n, float ta) : Klasse1(t, n)
19             { tall = ta;  sant = True;  }
20          bool ulik()  {  return sant;  }
21          void display()
22             { Klasse1::display(); cout << '\t' << tall << '\t' << sant <<
'\n'; }
23    };

24    int main() {
25      Klasse2 obj3("Emirates", 1, 1.0F), obj2("Stadium", 2, 1.8F);
26      Klasse1 obj1("Arsenal", 3);
27      obj1.display();  cout << '\t';
28      obj3.display();
29      obj2.display();
30      cout << obj3.lik("Arsenal") << '\t' << !obj1.lik("Arsenal") << '\t'
31           << obj2.ulik() << '\n';
32      return 0;
33    }
```

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

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

char txt[] = "ARSENAL-SLO-LIVERPOOL-IGJEN!!!";

void funk(int a, int b=11, int c=4)
  {  for (int i = a;  i > b;  i-=c )  cout << txt[i];  }

void funk(int a, int b, char c)
  {  txt[a%3+b] = txt[a%4+b] = txt[a%5+b]= c;  }

int main()  {
  char*  t = &txt[9];

  for ( ; *t != '-'; t+=3) cout << *t;   cout << '\n';
  funk(27, 12, 5);         cout << '\n';
  funk(22);                cout << '\n';
  funk(25, 12, '_');  t = &txt[12]; t[9] = '\0';   cout << t << '\n';
  funk(15, 2, '$');   txt[7] = '\0';               cout << txt << '\n';
  return 0;
}
```

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

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

class A  {
protected:
    int a3, a1;
  public:
    A(int a)                 {  a1 = 2*a/3;  a3 = a/4;  }
    virtual void display()   {  cout << a1 << ' ' << a3 << ' ';  }
    bool operator != (int a) {  return (a1 !=  a);  }
    int  operator * (int a)  {  return (a3 * a);  }
};


class B : public A  {
  private:
    int a1, b2;
  public:
    B(int a3, int b) : A(b)  {  a1 = a3;  b2 = a3*b;  }
    void display()           {  A::display();  cout << a1 << ' ' <<
b2;  }
    int  operator + (int a)  {  return (a1+b2+a3+a1+a); }
};


int main()  {
  A* aob  = new A(19);     aob->display();    cout << '\n';
  B* bob  = new B(23, 3);  bob->display();    cout << '\n';
  A* aob2 = new A(36);     cout << ((*aob2) != 9) << ' '
                                << ((*aob2) * 4) << '\n';
  A* aob3 = new B(9, 5);   aob3->display();   cout << '\n';
  B* bob2 = new B(9, 5);   cout << ((*bob2 + 7)) << '\n';
  return 0;
}
```

Oppgave 2 (70%)

Les *hele* teksten for denne oppgaven *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, `enum`'en, de fire klassene med datamedlemmer og (ferdiglagde) funksjoner, global variabel, `main()` og de tre ferdiglagde `les(…)`–funksjonene.
Bruk *alt* dette *svært* aktivt!

På skolen har du en ikke fullt så datakyndig lærer, som elsker å samle på gamle gjenstander/antikviteter. Han ønsker seg svært gjerne et lite program som holder orden på gjenstandene han har. Sakene kan deles inn i tre kategorier: malerier, møbler og ting (f.eks. glass, kopper, asjetter, gafler, kniver).

Datastrukturen

Datastrukturen består (som du ser i vedlegget) av *en* liste (`gjenstandene`) med *ulike typer* (ikke-homogene) objekter. *Alle klasser, det de arver, alle data-medlemmer og alle prototyper for medlemsfunksjoner er komplett og fullstendig deklarert/definert i vedlegget*. Din oppgave blir å skrive innmaten til funksjoner (totalt 23 stk.) inni og utenfor klassene.

Oppgaven

a) **Skriv innmaten til funksjonen** `Type les()`

Brukeren blir bedt om (vha. aktuell `les`) å skrive *en* bokstav (A(maleri), O(Møbel), T(ting)). Ut fra tegnet (loop evt. for å sikre at det er en av de lovlige) returneres aktuell `enum`-verdi.

b) **Skriv innmaten til funksjonene** `void skriv_en_gjenstand()` **og de fire** `display()` **-funksjonene inni klassene**

Den første funksjonen spør brukeren om navnet til en gjenstand. Om denne finnes, skrives (vha. relevante `display`-funksjoner) alle dens data ut på skjermen – ellers kommer det en melding.

c) **Skriv innmaten til funksjonen** `void skriv_alle_av_type()`

Funksjonen spør først om en type/kategori gjenstand (vha. funksjonen laget i oppgave 2a). Deretter gås det manuelt gjennom hele listen, og alle gjenstandene som er av denne typen/ kategorien skrives ut på skjermen. Bruk den ferdiglagde == -funksjonen.

d) **Skriv innmaten til funksjonene** `void ny_gjenstand()` **og de fire constructorene som har *en* parameter**

Den første funksjonen spør om den nye gjenstandens navn. Finnes denne allerede i listen, kommer det en melding. I motsatt fall leses en gjenstandstype (vha. funksjonen laget i oppgave 2a), og et nytt relevant objekt opprettes og legges inn i listen.

De fire constructorene sørger for at alle datamedlemmer fylles/settes (bruk aktuelle `les`'er og `const`'er). Husk også å oppdatere datamedlemmet `type`.

e) **Skriv innmaten til funksjonen** `void slett_gjenstand()`

Først leses en gjenstands navn. Finnes *ikke* denne, kommer det en melding. I motsatt fall spørres brukeren om hun/han *virkelig* ønsker å slette denne. Om svaret er "ja", utføres en sletting, og en melding om utført sletting kommer. Ellers skrives en melding om at ingenting ble slettet.

f) **Skriv innmaten til de fem** `void skriv_til_fil(…)` **-funksjonene.**

Funksjonene skal sørge for at *hele* datastrukturen skrives til filen "GJENSTANDER.DTA". Formatet bestemmer du helt selv, men **det skal oppgis som en del av besvarelsen**.

g) **Skriv innmaten til funksjonene** `void les_fra_fil()` **og de fire constructorene som har *to* parametre**

Funksjonene skal til sammen sørge for at *hele* datastrukturen blir lest inn igjen fra filen angitt ovenfor, og med det formatet du selv der bestemte.

Annet (klargjørende?):

- Du *skal* bruke LISTTOOL ifm. løsningen av denne oppgaven.
- Legg spesielt merke til:
  - den ferdiglagde ==-funksjonen inni klassen `Gjenstand`.
  - at datamedlemmet `type` er protected i klassen `Gjenstand`.
  - den ferdiglagde programsetningen ifm. `case 'A':` i `main`.
- Gjør dine egne forutsetninger og presiseringer av oppgaven, dersom du skulle finne dette nødvendig. Gjør i så fall klart rede for disse *i starten* av din besvarelse av oppgaven.

Lykke til – og god antikvitetsauksjon! ☺

frode_ætt_haugianerne.no

Vedlegg 1: Halvferdig programkode

```
                        //  INCLUDE:
#include <iostream>           //  cin, cout
#include <fstream>            //  i(f)stream, o(f)stream
#include <cstring>            //  strlen, strcpy
#include <cctype>             //  toupper
#include <cstdlib>            //  atof
#include "listtool.h"         //  "Verktøykasse" for listehåndtering.
using namespace std;

                        //  CONST og ENUM:
const int  STRLEN =     80;   //  Standard streng-/tekstlengde.
const int  MINAAR =   1500;   //  Gjenstand ikke eldre enn dette.
const int  MAXAAR =   2030;   //  Gjenstand ikke yngre enn dette.
const int  MINANT =      1;   //  Min. antall av en gjenstand (jfr. Ting).
const int  MAXANT =    100;   //  Max. antall av en gjenstand (jfr. Ting).
const int  MINSUM =      1;   //  Min. verdi/betalt for en gjenstand.
const int  MAXSUM = 200000;   //  Max. verdi/betalt for en gjenstand.
enum  Type { maleri, mobel, ting  };     //  Aktuelle subklasse-typer.

                        //  KLASSER:
class Gjenstand : public Text_element  {
  private:                    //  Text = navn
    int   aar;                //  Gjenstanden er fra dette året (ca.)
    float verdi, betalt;      //  Dens (antatte ca.) verdi og kjøpesum.
  protected:
    Type  type;               //  Brukes/settes av subklassene.
  public:
    Gjenstand(char* t);                          //  Lag innmaten ifm. oppgave 2D.
    Gjenstand(char* t, istream* inn);            //  Lag innmaten ifm. oppgave 2G.
    virtual void display();                      //  Lag innmaten ifm. oppgave 2B.
    bool operator == (Type typ)  {  return (type == typ);  }
    virtual void skriv_til_fil(ostream* ut);     //  Lag innmaten ifm. oppgave 2F.
};

class Maleri : public Gjenstand  {
  private:
    char* kunstner;                              //  Kunstnerens navn.
  public:
    Maleri(char* t);                             //  Lag innmaten ifm. oppgave 2D.
    Maleri(char* t, istream* inn);               //  Lag innmaten ifm. oppgave 2G.
    ~Maleri()  { delete [] kunstner;  }
    void display();                              //  Lag innmaten ifm. oppgave 2B.
    void skriv_til_fil(ostream* ut);             //  Lag innmaten ifm. oppgave 2F.
};

class Mobel: public Gjenstand  {
  private:
    char* beskrivelse;                           //  Beskrivelse av møbelet.
  public:
    Mobel(char* t);                              //  Lag innmaten ifm. oppgave 2D.
    Mobel(char* t, istream* inn);                //  Lag innmaten ifm. oppgave 2G.
    ~Mobel()  { delete [] beskrivelse;  }
    void display();                              //  Lag innmaten ifm. oppgave 2B.
    void skriv_til_fil(ostream* ut);             //  Lag innmaten ifm. oppgave 2F.
};

class Ting : public Gjenstand  {
  private:                        //  Materialet tingen er laget av:
    char* stoff;                  //     porselen, krystall, stål, jern,...
    int   antall;                 //  Antall duplikater man har av gjenstanden.
  public:
    Ting(char* t);                               //  Lag innmaten ifm. oppgave 2D.
    Ting(char* t, istream* inn);                 //  Lag innmaten ifm. oppgave 2G.
    ~Ting()  { delete [] stoff;  }
```

```
    void display();                             //  Lag innmaten ifm. oppgave 2B.
    void skriv_til_fil(ostream* ut);            //  Lag innmaten ifm. oppgave 2F.
};
```

```
                        //  DEKLARASJON AV FUNKSJONER:
void skriv_meny();
char  les(const char* t);
float les(const char* t, const int MIN, const int MAX);
void  les(const char t[], char s[], const int LEN);
Type  les();                                    //  Lag innmaten ifm. oppgave 2A.
void skriv_en_gjenstand();                      //  Lag innmaten ifm. oppgave 2B.
void skriv_all_av_type();                       //  Lag innmaten ifm. oppgave 2C.
void ny_gjenstand();                            //  Lag innmaten ifm. oppgave 2D.
void slett_gjenstand();                         //  Lag innmaten ifm. oppgave 2E.
void skriv_til_fil();                           //  Lag innmaten ifm. oppgave 2F.
void les_fra_fil();                             //  Lag innmaten ifm. oppgave 2G.


                        //  GLOBALE VARIABLE:
List* gjenstandene;                     //  Liste med ALLE de ulike gjenstandene.



int main()  {           //  HOVEDPROGRAM:
  char valg;

  gjenstandene = new List(Sorted);                          //  Initierer listen.

  les_fra_fil();                                            //  Oppgave 2G

  skriv_meny();
  valg = les("\n\nKommando");
  while (valg != 'Q')  {
    switch(valg)  {
      case 'A': gjenstandene->display_list();  break;
      case 'E': skriv_en_gjenstand();          break;          //  Oppgave 2B
      case 'T': skriv_all_av_type();           break;          //  Oppgave 2C
      case 'N': ny_gjenstand();                break;          //  Oppgave 2D
      case 'S': slett_gjenstand();             break;          //  Oppgave 2E
      default:  skriv_meny();                  break;
    }
    valg = les("\n\nKommando");
  }
  skriv_til_fil();                                          //  Oppgave 2F
  cout << "\n\n";
  return 0;
}



// **************************  Gjenstand:  **************************
Gjenstand::Gjenstand(char* t) : Text_element(t)  {
      //  Oppgave 2D:  Lag innmaten
}


Gjenstand::Gjenstand(char* t, istream* inn) : Text_element(t)  {
      //  Oppgave 2G:  Lag innmaten
}


void Gjenstand::display()  {
      //  Oppgave 2B:  Lag innmaten
}


void Gjenstand::skriv_til_fil(ostream* ut)  {
      //  Oppgave 2F:  Lag innmaten
}
```

```
// ******************************  Maleri:  *********************************
Maleri::Maleri(char* t) : Gjenstand(t)  {
      //  Oppgave 2D:  Lag innmaten
}

Maleri::Maleri(char* t, istream* inn) : Gjenstand(t, inn)  {
      //  Oppgave 2G:  Lag innmaten
}

void Maleri::display()  {
      //  Oppgave 2B:  Lag innmaten
}

void Maleri::skriv_til_fil(ostream* ut)  {
      //  Oppgave 2F:  Lag innmaten
}

// ******************************  Møbel:  **********************************
Mobel::Mobel(char* t) : Gjenstand(t)  {
      //  Oppgave 2D:  Lag innmaten
}

Mobel::Mobel(char* t, istream* inn) : Gjenstand(t, inn)  {
      //  Oppgave 2G:  Lag innmaten
}

void Mobel::display()  {
      //  Oppgave 2B:  Lag innmaten
}

void Mobel::skriv_til_fil(ostream* ut)  {
      //  Oppgave 2F:  Lag innmaten
}

// ******************************  Ting:  ***********************************
Ting::Ting(char* t) : Gjenstand(t)  {
      //  Oppgave 2D:  Lag innmaten
}

Ting::Ting(char* t, istream* inn) : Gjenstand(t, inn)  {
      //  Oppgave 2G:  Lag innmaten
}

void Ting::display()  {
      //  Oppgave 2B:  Lag innmaten
}

void Ting::skriv_til_fil(ostream* ut)  {
      //  Oppgave 2F:  Lag innmaten
}


void skriv_meny()  {               //  Skriver alle mulige menyvalg:
  cout << "\n\nFØLGENDE KOMMANDOER ER TILGJENGELIGE:"
       << "\n   A  - skriv Alle gjenstandene"
       << "\n   E  - skriv alt om En spesiell gjenstand"
       << "\n   T  - skriv gjenstander av en viss Type (maleri, møbel, ting)"
       << "\n   N  - Ny gjenstand"
       << "\n   S  - Slett/fjern en gjenstand"
       << "\n   Q  - Quit / avslutt";
```

```
}
```

```
char les(const char* t)  {          //  Leser og upcaser brukerens valg/ønske:
  char ch;
  cout << t << ": ";
  cin >> ch;   cin.ignore();
  return (toupper(ch));
}

                                    //  Leser en FLOAT mellom MIN og MAX:
float les(const char* t, const int MIN, const int MAX)  {
  char text[STRLEN];
  float n;
  do  {
    cout << '\t' << t << " (" << MIN << '-' << MAX << "): ";
    cin.getline(text, STRLEN);   n = atof(text);     // Leser som tekst - omgjør:
  } while (n < MIN  ||  n > MAX);
  return n;
}

                                    //  Leser inn en ikke-blank tekst:
void les(const char t[], char s[], const int LEN) {
  do  {
    cout << '\t' << t << ": ";      //  Skriver ledetekst.
    cin.getline(s, LEN);            //  Leser inn tekst.
  } while (strlen(s) == 0);         //  Sjekker at tekstlengden er ulik 0.
}


Type les()  {                       //  Leser og returnerer lovlig type gjenstand:
      // Oppgave 2A:  Lag innmaten
}


void skriv_en_gjenstand()  {        // Skriv EN spesiell gjenstand:
      // Oppgave 2B:  Lag innmaten
}


void skriv_all_av_type()  {         //  Skriv EN spesiell gjenstand:
      // Oppgave 2C:  Lag innmaten
}


void ny_gjenstand()  {              //  Legg inn en ny gjenstand:
      // Oppgave 2D:  Lag innmaten
}


void slett_gjenstand()  {           //  Slett/fjern en gjenstand:
      // Oppgave 2E:  Lag innmaten
}


void skriv_til_fil()  {             //  Skriv HELE datastrukturen til fil:
      // Oppgave 2F:  Lag innmaten
}


void les_fra_fil()  {               //  Leser HELE datastrukturen fra fil:
      // Oppgave 2G:  Lag innmaten
}
```