TENTAMEN
DVA222 - OBJEKTORIENTERAD PROGRAMMING

19 Aug 2015 kl. 8:10–13:30
Afshin Ameri, 021-101550

- Denna tentamen omfattar 7 uppgifter och totalt 42 poäng.
- Betygsgränser: 21p: 3, 30p: 4, 37p: 5
- ECTS betygsgränser: 21p: D, 26p: C, 31p: B, 37p: A
- Alla svar skall motiveras och om förutsättningar saknas skall rimliga antaganden göras.
- ***Inga hjälpmedel är tillåtna.***
- ***Påbörja varje uppgift på ett nytt papper!***

Lycka Till,
Afshin

1- ALLMÄNT - GENERAL(6P)

SWE: Förklara och exemplifiera följande reserverade ord (keywords) i C++:
ENG: Describe the following reserve words (keywords) in C++ language. Use examples:

- `friend`
- `static`
- `catch`
- `delete`
- `inline`
- `virtual`

2- KLASSER - CLASSES(6P)

SWE:
(a) Tänk dig att du utvecklar en del av ett program där användaren skall mata in ett egenvalt lösenord. Följande krav ska ställas på det inmatade lösenordet:

- Det skall vara minst 7 tecken långt.
- Det måste innehålla minst ett siffertecken.
- Det måste innehålla minst en stor bokstav.
- Det måste innehålla minst en liten bokstav.
- Det får inte innehålla några mellanslag.

Skapa en hjälpklass `PasswordCriteriaValidator` som kan användas för att verifiera huruvida en given textsträng som innehåller det inmatade lösenordet, uppfyller ovanstående krav eller ej. Om kraven ej uppfylls, så skall orsaken returneras.

(b) Skriv ett program som testar klassen `PasswordCriteriaValidator`. Låt programmet fråga användaren efter ett lösenord. Använd sedan din klass för att validera lösenordet. Om kriterierna ovan ej är uppfyllda ska programmet skriva ut vad som är problemet och fråga efter ett nytt lösenord.

ENG:
(a) Imagine that as part of a development team you are responsible for developing a part of a program where the user should choose a password. The chosen password should fulfill the following criteria:

- In must be at least 7 characters long.
- It must at least contain one digit.
- It must at least contain one upper case letter.
- It must at least contain one lower case letter.
- It should not have any whitespace characters.

Implement a helper class `PasswordCriteriaValidator` that can be used to verify if a given text string fulfills the password requirements or not. If not, the reason should be returned.

(b) Implement a program that tests the `PasswordCriteriaValidator` class. The program asks the user for a new password and then uses the helper class to validate the password. If the validation fails, the program should print out the reason and ask for a new password.

3- ARV - INHERITANCE (6P)

SWE: I ett program som ska simulera fritidsbåtar skall följande olika typer av båtar finnas:

- Motorbåt (fabrikat, längd, djupgående, motorstyrka, ägare)
- Segelbåt (fabrikat, längd, djupgående, masthöjd, ägare)
- Roddbåt (fabrikat, längd, ägare)
- Motorseglare (fabrikat, längd, djupgående, motorstyrka, masthöjd, ägare)

Inom parentes har de datafält (instansvariabler) som behövs för varje båttyp specificerats. Skapa en lämplig arvshierarki för dessa båtar. Lägg märke till gemensamma datafält i specifikationen och utnyttja dessa i din arvshierarki. Du skall definiera klasserna i C++. Det räcker dock om du skriver in alla instansvariablerna i klasserna, samt lämpliga konstruktorer. Du behöver således inte implementera några metoder i klasserna.

ENG: In a program which is designed to simulate boats, the following types of boats are represented:

- Motor Boat (manufacturer, length, depth, motor power, owner)
- Sail Boat (manufacturer, length, depth, mast length, owner)
- Rowing Boat (manufacturer, length, owner)
- Motor Sail Boat (manufacturer, length, depth, motor power, mast length, owner)

The data fields (instance variables) required for each boat type are given in the parenthesis in front of them. Create a class hierarchy for these boat types. Pay attention to common data fields and use them in your class hierarchy. The classes must be defined in C++. It is enough to write all the instance variables and related constructors in the classes. You do not need to implement any methods in the classes.

4- DYNAMISK BINDNING (6P)

Vad skrivs ut när nedanstående program körs? Motivera ditt svar!

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

class A {
public:
    A() {cout << "A ctor" << endl;}
    A(A& a) {cout << "A copy ctor" << endl;}
    virtual ~A() {cout << "A dtor" << endl;}
    virtual void f() {cout << "A f" << endl;}
};

class B : public A {
public:
    B() {cout << "B ctor" << endl;}
    ~B() {cout << "B dtor" << endl;}
    void f() {cout << "B f" << endl;}
};

class C : public B {
public:
    C() {cout << "C ctor" << endl;}
    ~C() {cout << "C dtor" << endl;}
    void f() {cout << "C f" << endl;}
};

class X {
public:
    X() {cout << "X ctor" << endl;}
    X(X& x) {cout << "X copy ctor" << endl;}
    ~X() {cout << "X dtor" << endl; }
    void f() {cout << "X f" << endl; }
};

X foo(A& a, X x)
{
    cout << "*** in foo" << endl;
    A aa;
    aa = a;
    X xx(x);
    cout << "*** leaving foo" << endl;
    return xx;
}

int main() {
    cout << "*** starting main" << endl;
    A * a_ptr_to_a = new A;
    B * b_ptr_to_b = new B;
    C * c_ptr_to_c = new C;
    X x1;
    X * x_ptr_to_x = &x1;
    A * a_ptr_to_b = b_ptr_to_b;
    A * a_ptr_to_c = c_ptr_to_c;
    B * b_ptr_to_c = c_ptr_to_c;
```

```
    cout << "*** calls of f #1:" << endl;
    a_ptr_to_a -> f();
    b_ptr_to_b -> f();
    c_ptr_to_c -> f();
    x_ptr_to_x -> f();

    cout << "*** calls of f #2:" << endl;
    a_ptr_to_b -> f();
    a_ptr_to_c -> f();
    b_ptr_to_c -> f();

    X x2;

    cout << "*** calling foo" << endl;
    x2 = foo( *a_ptr_to_a, x1 );

    cout << "*** back in main" << endl;
    cout << "*** delete a_ptr_to_a:" << endl;
    delete a_ptr_to_a;

    cout << "*** delete b_ptr_to_b:" << endl;
    delete b_ptr_to_b;

    cout << "*** delete b_ptr_to_c:" << endl;
    delete b_ptr_to_c;

    cout << "*** leaving main" << endl;
    return 0;

}
```

5- OPERATORÖVERLAGRING - OPERATOR OVERLOADING (6P)

SWE: Klassen `Date` som används för att representera datum har definierats enligt följande:
ENG: Class `Date` is deigned to represent a date as following:

DATE.H

```
class Date{
private:
    static int daysInMonth[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
    int year, month, day;

    bool isLeapYear();
    int getDaysInMonth();
    void validate();

public:
    Date(int y, int m, int d);
    int GetYear(){ return year; }
    int GetMonth(){ return month; }
    int GetDay(){ return day; }
    void Set(int y, int m, int d);
};
```

DATE.CPP

```
#include "date.h"

bool Date::isLeapYear(){
    return((Year % 4 == 0 && Year % 100 != 0) || Year % 400 == 0);
}
int Date::getDaysInMonth(){
    if (Month == 2 && isLeapYear()) { return 29; }
    else return daysInMonth[Month];
}
void Date::validate(){
    if (year < 1) year = 1;
    if (month < 1 || Month > 12) month = 1;
    if (day < 1 || day > getDaysInMonth()) day = 1;
}
Date::Date(int y, int m, int d){
    Set(y,m,d);
}
Date::Set(int y, int m, int d){
    year = y; month = m; day = d;
    validate();
}
```

SWE: Ändra klassen så att det kan stödja följande operationer (Använd dig av operatoröverlagring) :
ENG: Change the class in a way that it can support the following operations (use operator overloading) :

```
Date newDate(2015,8,19);
Date newerDate = newDate + 45;   //adds 45 days to the newdate
Date olderDate = newDate – 18;   //subtracts 18 days from newdate
Date diffDays = newerDate – olderDate; //return the difference

cout << olderDate;
cout << newerDate;
```

6- GENERISKA CONSTRUKTIONER - GENERIC CONSTRUCTORS(6P)

SWE: Vad är en generisk klass (template)? Hur man använder den? Ge ett exempel.

ENG: What is a generic class (template)? How do we use them? Give an example.

7- OBJEKTORIENTERAD ANALYS OCH DESIGN – OBJECT ORIENTED ANLYSIS AND DEISGN (6P)

SWE: ”Encapsulation”, ”Polymorphism” och ”Inheritance” är de tre viktigaste designprinciperna i objektorienterad programmering. Förklara och exemplifiera dem.

ENG: “Encapsulation”, “Polymorphism” and “Inheritance” are three important design principles of object oriented programming. Describe each of them using examples.