**Innlevering Datamodellering april 2017**

**Spørsmål 1**

Første del av spørsmål 1 (a og b) er tidlegare gitt som øvingsoppgåve (15. mars) med fasit. Så den vil ikkje bidra til karakteren på innleveringa.

Kystvakten skal lage ein oversikt over alle fartøy som driv rutetransport på kysten. Dei klassifiserer fartøy i 3 typar: Lastebåt som ikkje tar passasjerar eller bilar, ferje som ikkje tar last og snøggbåtar som berre tar passasjerar.

Ein første ”kladd” for ein databasetabell ser slik ut utfylt med eksempeldata:

DATA

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fartøy | Rederi | Havner | Type | Bilar | Pas. | Last | Rederitlf | Kallesign. |
| Stord | Tide | Haugesund, Leirvik, Bergen | Last | 0 | 0 | 700 | 555 05050 | LD342 |
| Florø | Fjord1 | Halgjem, Sandvikvåg | Ferje | 150 | 450 | 0 | 575 05050 | HX154 |
| Varg | Fjord1 | Bergen, Haugesund, Stavanger | Snøggbåt | 0 | 150 | 0 | 575 05050 | WM072 |

Ingen fartøy kan ha samme namn eller samme kallesignal. Havner viser kva rute fartøyet for tida går.

1. Analyser denne tabellen ved å skissere funksjonelle determineringar og kanditatnøklar og bestemme normalform. Konstruer så normalisert(e) tabell(ar) for dataene. Grunngi kvifor kvar tabell er normalisert.
2. Skisser ein UML-modell for datasystemet. Bruk entiteter: Havn, Fartøy, Rederi, Ferje, Snøggbåt og Lastebåt.
3. For vidare bruk av dataene skal det lagast eit C++ - program, som byggjer på UML-modellen.

Klassen Rederi er definert nedanfor:

//rederi.h  
#include <string>  
#include <iostream>

using namespace std;

class Rederi{

public:

Rederi(); //Konstruktør utan argument

void settNamn(string NyttRederinamn);

void settTlf(string Nyttlf);

string finnRederinamn() const;

string finnTlf() const;

private:

string rederinamn;

string tlf;

};

Definer klassane Fartoy og Snoeggbaat. Vi tar ikkje med entiteten Havner i programmet. Fartoy skal vera ein abstrakt klasse med ein ekte virtuell funksjon SkrivData(). I klassen Snoeggbaat skal SkrivData() lage ei slik utskrift (der ……står for data):

Snøggbåten ..........med kallesignal …….. tar …….. passasjerar.  
Alle klassar skal ha default-konstruktørar.

Implementer alle funksjonane.

1. Lag ein C++ - applikasjon (eit hovedprogram main()) som definerer eit objekt R1 av klassen Rederi og eit objekt S1 av klassen Snoeggbaat. Applikasjonen skal settja inn dataene for Varg frå tabellen (unntatt havner). Så skal applikasjonen skrive ut dataene slik:

Snøggbåten *Varg* med kallesignal *WM072* tar *450* passasjerar og tilhøyrer *Fjord1*

Det som står i kursiv skal hentast frå objektet S1.

Applikasjonen skal også definere eit objekt S2 av klassen Snoeggbaat der alle data er default-verdiar, og skrive ut data for denne snøggbåten.

**Spørsmål 2**

Klassen Kompleks er gitt med to filer som ligg i katalogen . Vi skal lese inn komplekse tal i ein tabell, sortere dei og skrive ut den sorterte tabellen. (Normalt kan vi ikkje sortere komplekse tal, men sidan vi har definer operatoren < , kan vi her sortere dei etter absoluttverdi.) Programma skal innehalde ein funksjon ***Lag()*** som spør etter realdel og imaginærdel og returnerer eit kompleks tal, og ein funksjon ***SkrivUt()*** for utskrift av eit kompleks tal.

1. Lag eit program **KompleksTabell.cpp** til denne jobben. Alle variablane i hovedprogrammet skal ligge i det frie lageret. Start med å spørje kor mange tal tabellen skal innehalde.
2. Lag eit program ***KompleksVector.cpp*** som utfører den samme jobben. Programmet skal bruke **STL-vector og –algoritmer** til sorteringa.
3. Lag eit program ***KompleksTemplate.cpp*** til den samme jobben. Programmet skal bruke templaten ***diverse*** i TemplateSort.cpp til sorteringa.