Eksamensopgave i RD2-SWC

1. juli 2022

Alle offline hjælpemidler er tilladt.

Aflevering

Der skal afleveres én samlet PDF med opgavebesvarelsen af både C++-delen og softwareudviklingsdelen, som alene er det dokument, der bliver bedømt. I kan kun aflevere et dokument som altså både skal indeholde C++ afleveringen med kildekode (formateret som farvet tekst) og test samt softwareudviklingsdelen med SQL-kode (formateret som tekst).

Til hver C++ opgave er der en ***test-kode***, som skal ***eksekveres*** og et ***output/screendump*** skal indsættes i afleveringen. Hvis programmet ikke kan compile, så indsættes i stedet ***fejlmeddelelsen*** fra compileren. Generelt vurderes opgaverne ud fra en korrekt anvendelse af de tilegnede C++ færdigheder, som er opnået gennem gennemførelse af faget. Det forventes, at de tillærte færdigheder anvendes når det kan, herunder anvendelsen af referencer ved ikke simple datatyper samt brug af const, hvor det er muligt.

Enhver form for kopier/indsæt (copy/paste) fra tidligere opgaver eller andre kilder anses som eksamenssnyd. Kode fra denne eksamensopgave må dog gerne kopieres ind.

Opgavesættet består af 7 sider, 1 forside, 4 sider med C++ opgaver og 2 sider med SQL og softwareudviklingsopgaver.

Opgave 1 (45 minutter, 20 point)

I denne opgave skal vi kigge på en tabel med overskrifter.

```
class Table
{
public:
    Table() {}

    void setHeaders(std::string h1, std::string h2, std::string h3);
    void addData(int i, std::string s, double d);

    void outputData(std::ostream& os);

private:
    std::tuple<std::string, std::string, std::string> mHeaders;
    std::vector<std::tuple<int, std::string, double> > mData;
};
```

Includes: ostream, tuple og vector

Implementer klassen Table

I første del af opgaven skal klassen Table implementeres færdig. Alle funktioner er givet i headeren, men SKAL implementeres i en cpp-fil og opfylde følgende krav:

- Der skal laves en funktion setHeaders, der som input tager tre strenge. Funktionen skal sætte membervariablen mHeaders tre elementer. mHeaders er en std::tuple.
- Funktionen addData skal føje data til member variablen mData. mData er implementeret som en vector, der indeholder tuples med datatyperne int, string og double. addData funktionen tager 3 argumenter, netop disse tre elementer af typen int, string og double.
- Funktionen outputData skal udskrive en tabel til parameteren *os*. Formatet skal følge nedenstående
    -
```
    H1 H2                              H3
-------------------------------------
    i s                                 d
```
    - Her er H1, H2 og H3 de tre overskrifter som er gemt i mHeaders.
    - Elementer fra mData skrives på en ny linje, hvor i er heltallet, s er strengen og d er kommatallet. For hvert dataelement skal der altså udskrives en sådan linje.
    - Der benyttes højrejustering i første søjle, venstrejustering af teksterne og højrejustering af kommatallene.
    - Kommatallene udskrives altid med to cifre efter kommaet.
    - Der skal anvendes 5 tegn til første søjle, herefter et mellemrum, så anvendes 20 tegn til teksten efterfulgt af 10 tegn til kommatallet
    - Eksempel på output kunne være:

```
Antal Tekst                       Beløb
-------------------------------------
    1 Jordbær                     15.00
    2 Elektronik                  99.99
    1 Maske                      150.00
    3 Løg                         10.90
```

Test-kode (HUSK Screendump af test)

```
    std::cout << std::endl << std::endl;
    std::cout << "--------------" << std::endl;
    std::cout << "-- Opgave 1 --" << std::endl;
    std::cout << "--------------" << std::endl;

    Table t;
    t.setHeaders("Ref", "Tekst", "Kontant");
    t.addData(1, "Sko", 1025.25);
    t.addData(2, "Toj", 699.99);
    t.addData(1, "Taske", 1500.00);
    t.addData(1, "Andet", 249.90);
    t.addData(1, "Toj", 530.15);
    t.addData(2, "Sko", 799.00);
    t.addData(2, "Andet", 225.40);
    t.outputData(std::cout);
    std::cout << std::endl << std::endl;
```

Opgave 2 (1 time og 15min, 30 point)

I denne opgave skal vi kigge på klassen DateTime:

```
class DateTime
{
public:
    DateTime() {}

    DateTime(int year, unsigned int month, unsigned int day, unsigned int hour,
unsigned int minute, unsigned int second);

private:
    int mYear = 1970;
    unsigned int mMonth = 1, mDay = 1;
    unsigned int mHour = 0, mMinute = 0, mSecond = 0;
};
```

Includes: ostream

Følgende skal implementeres på/til klassen DateTime (implementationerne SKAL være i en separat cpp-fil)

- Constructoren som tager alle parametrene year, month, day, hour, minute og second. Constructoren SKAL bruge en initializer list og sætte de tilsvarende member variable.
- Sammenligningsoperatoren ”mindre end” (<) som const-funktion
- Sammenligningsoperatoren ”større end” (>) som const-funktion
- Sammenligningsoperatoren ”lig med” (= =) som const-funktion
- Insertion operatoren (operator<<) skal implementeres som friend til klassen. Output skal være i formatet YYYY-MM-DD hh:mm:ss
    - YYYY angiver, at der altid bruges 4 tegn til årstallet
    - MM angiver, at der altid bruges 2 tegn til måneden
    - DD angiver, at der altid bruges 2 tegn til dagen
    - hh angiver, at der altid bruges 2 tegn til timetallet
    - mm angiver, at der altid bruges 2 tegn til minuttallet
    - ss angiver, at der altid bruges 2 tegn til sekundtallet.
    - ”-”, ” ”, ”:” angiver tegnene, der bruges mellem tallene.

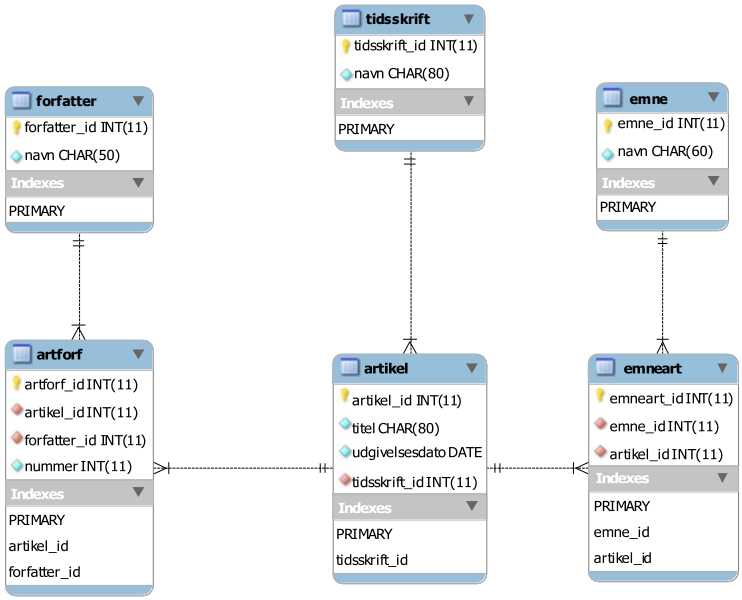
Test-koden findes på næste side.

Test-kode (HUSK Screendump af test)

```
    std::cout << std::endl << std::endl;
    std::cout << "--------------" << std::endl;
    std::cout << "-- Opgave 2 --" << std::endl;
    std::cout << "--------------" << std::endl;

    std::vector<DateTime> dtv;
    for (unsigned int y = 2000; y < 2005; ++y) {
        for (unsigned int m = y%2 + 1; m < 12; m += 3) {
            for (unsigned int d = (m+y)%3 + 1; d < 30; d += 5) {
                for (unsigned int h = d%2 + 7; h < 18 + m%2; h += d%2 + 1) {
                    for (unsigned int min = h%2; min < 60; min += d%2 + h%2 + 15) {
                        for (unsigned int s = min%2 + h%2 + d%2; s < 60; s += min%2 +
h%2 + d%2 + 18) {
                            dtv.emplace_back(y,m,d,h,min,s);
                        }
                    }
                }
            }
        }
    }


    std::ostream_iterator<DateTime> dti(std::cout, "\n");
    std::shuffle(dtv.begin(), dtv.end(), std::default_random_engine(20));

    std::vector<DateTime> dtv20(dtv.begin(), dtv.begin()+20);

    std::cout << std::endl << "Shuffled order" << std::endl;
    std::cout << "----------------" << std::endl;
    std::copy(dtv20.begin(), dtv20.end(), dti);
    std::sort(dtv20.begin(), dtv20.end());
    std::cout << std::endl << "Ascending order" << std::endl;
    std::cout << "----------------" << std::endl;
    std::copy(dtv20.begin(), dtv20.end(), dti);

    std::cout << std::endl << "Descending order" << std::endl;
    std::cout << "----------------" << std::endl;
    std::sort(dtv20.begin(), dtv20.end(), std::greater<DateTime>());
    std::copy(dtv20.begin(), dtv20.end(), dti);

    std::cout << std::endl << "Equality test" << std::endl;
    std::cout << "----------------" << std::endl;
    for (unsigned int i = 0; i < dtv20.size()-1; ++i) {
        std::cout << std::setw(2) << i << ": ";
        std::string test;
        test = dtv20[i] == dtv20[i] ? "YES" : "NO";
        std::cout << test << " ";
        test = dtv20[i] == dtv20[i+1] ? "YES" : "NO";
        std::cout << test << std::endl;
    }
```

Opgave 3 (SQL, 1 time, 25 point)

Opgaven tager udgangspunkt i nedenstående database og skal holde styr på forskningsartikler skrevet af medarbejdere ved en (fiktiv) forskningsenhed.



Artikler bliver offentliggjort i tidsskrifter og har en eller flere forfattere. Rækkefølgen i hvilken forfatterne er angivet i artiklen er vigtig, derfor attributten *nummer* i tabellen ***artforf***. Artikler vedrører et eller flere forskningsemner, og derfor er det nødvendigt med tabellen ***emneart***.

Databasen kan etableres i MySQL ved hjælp af tekstfilen ***ForskningsArtiklerDDL***.

I opgavebesvarelserne er det tilladt at spørge på indholdet af datafelter, men ikke på værdien af id'er.

FORFATTER.NAVN = 'Otto Fishbein' er således tilladt; men det er **FORFATTER.FORFATTER_ID = 5** *ikke*.

Opgave 1
Skriv en forespørgsel som kan vise, hvilke forskere (forfattere) der har fået offentliggjort en eller flere artikler i tidsskriftet *Robotics and automation*.

Opgave 2
Skriv en forespørgsel som kan vise, hvilke forskere der ikke har været førsteforfatter på en artikel.

Opgave 3
Skriv en forespørgsel som kan vise antallet af artikler, som *Henrik Salah Petersen* har været medforfatter på.

Opgave 4
Lav en liste over navnene på samtlige forfattere/forskere og det antal artikler, de har andel i. Sortér listen efter hvem der har medvirket i flest.

Opgave 5
Lav en liste over de emner som er behandlet i mindst tre artikler.

Opgave 4 (Domænemodellering, 1 time, 25 point)

Denne opgave går ud på at udarbejde en **domænemodel** til en 1. iteration af et it-system, som kan bruges til lokale- og kursusadministration for et gymnasium.

Gymnasiet er helt traditionelt og arbejder med elever, lærere, klasser, årgange, fag og lokaler. Systemet skal som minimum kunne opfylde følgende krav:

- Generering af skemaer for elever og lærere, dvs. angivelse af tidspunkter og lokale(r) for undervisningsaktiviteter.
- Tage højde for dobbeltbooking af lokaler og fag. Bestemt undervisning for elever i en klasse skal ligge på samme tidspunkt (fx 1. mx har altid matematik om onsdagen fra 10 til 11); et lokale må ikke være booket til flere undervisningsaktiviteter på samme tidspunkt etc.
- Give oplysninger om lokalers tilgængelighed. Fx oplyse hvornår et bestemt lokale er ledigt og/eller oplyse hvilke lokaler, der er ledige på et bestemt tidspunkt.
- Kunne oplyse om elevers 'whereabouts'; i hvilket lokale bør fx Lone Thomsen fra 1. sa være fredag kl. 10:30?

Det antages, at elever i samme klasse har identisk skema, dvs. begrebet valgfag eksisterer ikke og eksempelvis har alle i én klasse det/de samme sprogfag.

Du bestemmer selv, hvilke 'hjælpeartefakter' (brugsmønstre, sekvensdiagrammer, kontrakter), du vil lave. Husk at et artefakt kun skal udarbejdes, hvis det skaber værdi for systemet.

Det er imidlertid meget vigtigt, at du argumenterer for de trufne valg. Den gode besvarelse indeholder således relevante kommentarer til de indgående elementer (klasser, attributter og relationer mellem klasser).

Ikke alle kommentarer er lige relevante. Relevante kommentarer skal, i lighed med 'hjælpeartefakterne', skabe værdi enten i form af bedre forståelse for rationalet bag de trufne valg eller ved at forøge kendskabet til domænet.