Eksamensopgave i RD2-SWC

1. juni 2021

Alle hjælpemidler er tilladt. Husk at de generelle regler om snyd stadig gælder.

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 og test samt softwareudviklingsdelen.

Til hver C++ opgave er der en ***test-kode***, som skal ***eksekveres*** og et ***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. Det forventes, at der anvendes referencer, når der er tale om ikke-simple datatyper. Desuden forventes det, at const anvendes, 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 9 sider, 1 forside, 6 sider med C++ opgaver og 2 sider med software-udviklingsopgaven.

Opgave 1 (1-1½ time, 35 point)

I denne opgave skal der implementeres en simulering af en lodtrækning uden tilbagelægning. Altså en lodtrækning, hvor man har lodder i en pose eller spand, som trækkes en efter en indtil, der ikke er flere lodder tilbage. Dette kunne f.eks. være lodder med værdien 0 og lodder med værdien 1, hvor resultatet anvendes til en holdinddeling til en sportskamp, eksamen eller andet. På denne måde kan man, inden lodtrækningen sættes i gang, styre hvor mange, der får lodder med værdien 0 og hvor mange, som får lodder med værdien 1. Der kunne naturligvis også være flere forskellige værdier med forskellige fordelinger. I denne opgave vil vi dog kun koncentrere os om en ligelig fordeling mellem hver værdi. Hvis der f.eks. er 16 personer, som deltager i en lodtrækning med to udfald (0 eller 1), vil der være 8 personer, som trækker værdien 0 og 8 personer, som trækker værdien 1.

Til denne opgave skal vi bruge klassen "Bin", som repræsenterer den spand, vi trækker lodder fra. Deklarationen til denne klasse er givet herunder:

```
class Bin
{
public:
    Bin();

    void fillBin(int min, int max, int amount);

    int drawAndRemove();

    unsigned int count(int value) const;

    unsigned int size() const;

private:
    std::default_random_engine mRandomEngine;
    std::vector<int> mValues;
};
```

a) Implementer klassen Bin

I første del af opgaven skal klassen Bin implementeres færdig. Alle funktioner, som skal implementeres er givet i headeren og forklares herunder

- Constructor skal initialisere tilfældighedsgeneratoren, således programmet genererer en ny følge af tilfældige tal, hver gang programmet startes.
- Membervariablen *mValues* indeholder spandens lodder, som blot gemmes som heltalsværdier.
- Funktionen *fillBin* skal fylde tal i spanden (*mValues*), således der er en ligelig fordeling af hele tal mellem *min* og *max* (begge inklusive). Såfremt *max*-*min*+1 ikke går op i *amount* skal resten af tallene fordeles så jævnt som muligt.
- Funktionen *drawAndRemove*() skal trække et lod fra spanden (tilfældigt) og fjerne loddet efter det er trukket. Implementationen af denne funktion skal foretages effektivt og løsningen vurderes på baggrund af effektiviteten (HINT: Husk at loddernes rækkefølge i spanden er ligegyldig. HUSK også, at det er ineffektivt at fjerne i "starten" og "midten" af en vektor.)
- Funktionen *count* skal tælle antallet af lodder med værdien *value* i spanden.
- Funktion *size* skal angive antallet af lodder, der er tilbage i spanden.

b) Implementer klassen Student

I forbindelse med mange mundtlige eksamener, fordeler vi normalt de studerende (af hensyn til effektivitet) i to eller flere grupperinger. Hvis der f.eks. har været to undervisere på et kursus, vil fagligheden være trukket på forhånd. For at fordelingen bliver ligelig mellem de to fagligheder, vil der ofte være tale om en lodtrækning, som udføres uden tilbagelægning efter princippet i foregående delopgave. Denne delopgave går ud på at implementere en klasse *Student*, som repræsenterer en studerende, og som skal kunne udføre en enkelt lodtrækning vha. en spand med lodder (*Bin*).

```
class Student
{
public:
    Student();
    Student(const std::string& name, const std::string& username);

    void draw(Bin& bin);

    std::string getExam(const std::vector<std::string>& exams) const;

private:
    std::string mName;
    std::string mUsername;
    int mExam = -1;
};
```

Klassen til Student skal implementeres som ovenstående deklaration. Desuden skal der implementeres getters og setters efter de principper, som er undervist. Implementeringen skal udføres således:

- Constructors skal implementeres, hvor member variable sættes.
- Getters og setters *getName*, *setName* samt *getUsername* og *setUsername* skal implementeres.
- Funktionen *draw* skal udføre selve lodtrækningen vha. parameteren Bin. Dvs. *draw*-funktionen skal altså anvende *bin* til at udføre lodtrækningen. Resultatet af lodtrækningen gemmes i member variablen *mExam* som et tal.
- Funktionen *getExam* tager en vector med strenge. F.eks. kunne vector'en indeholde {"PLC", "OOP"} som et eksempel. Lodtrækningen (gemt i *mExam*) vil således repræsentere indeks i denne vektor, dvs. 0 vil betyde PLC og 1 vil betyde OOP i netop dette tilfælde.

c) Implementer insertion operator (<<)

På klassen *Student* skal der implementeres en insertion operator (<<).

Se test-kode for opgave 1 på næste side (HUSK screendump af testen):

Test-kode

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

    // Test lodtrækning uden tilbagelægning
    Bin b;
    b.fillBin(1,2,10);
    std::cout << "Bin size (before): " << b.size() << std::endl;
    std::cout << "Number of 1's: " << b.count(1) << std::endl;
    std::cout << "Number of 2's: " << b.count(2) << std::endl;
    std::cout << "Draw : ";
    for (unsigned int i = 0; i < 10; ++i) {
        std::cout << b.drawAndRemove() << " ";
    }
    std::cout << std::endl;
    std::cout << "Bin size (after): " << b.size() << std::endl;
    std::cout << std::endl;


    b.fillBin(2,4,10);
    std::cout << "Bin size (before): " << b.size() << std::endl;
    std::cout << "Number of 2's: " << b.count(2) << std::endl;
    std::cout << "Number of 3's: " << b.count(3) << std::endl;
    std::cout << "Number of 4's: " << b.count(4) << std::endl;
    std::cout << "Draw : ";
    for (unsigned int i = 0; i < 10; ++i) {
        std::cout << b.drawAndRemove() << " ";
    }
    std::cout << std::endl;
    std::cout << "Bin size (after): " << b.size() << std::endl;
    std::cout << std::endl;



    std::cout << std::endl;
    std::cout << "--------------------" << std::endl;
    std::cout << "-- Opgave 1b test --" << std::endl;
    std::cout << "--------------------" << std::endl << std::endl;

    // Test lodtrækning af faglighed ved en eksamen
    std::vector<std::string> exams1 {"PLC", "OOP"};
    std::vector<std::string> exams2 {"IIC", "RSI", "VIS"};
    std::vector<Student> students;
    students.emplace_back("Hans Hansen", "hahan20");
    students.emplace_back("Hanne Hansen", "haha120");
    students.emplace_back("Jens Jensen", "jejen20");
    students.emplace_back("Jesper Jespersen", "jejes20");
    students.emplace_back("Henrik Henriksen", "hehen20");
    students.emplace_back("S\x9Bren S\x9Brensen", "sosor20");
    students.emplace_back("Niels Nielsen", "ninie20");
    students.emplace_back("Peder Pedersen", "peped20");
    students.emplace_back("Anders Andersen", "anand20");
    students.emplace_back("Christian Christensen", "chchr20");
    students.emplace_back("Lars Larsen", "lalar20");
    students.emplace_back("Rasmus Rasmussen", "raras20");
    students.emplace_back("J\x9Brgen J\x9Brgensen", "jojor20");

    std::cout << "Studerende og brugernavne" << std::endl;
    for (Student& s : students) {
        std::cout << std::left << std::setw(22) << s.getName() << " " <<
s.getUsername() << std::endl;
    }
```

```
    std::cout << std::endl;

    // Lodtrækning uden tilbagelægning
    Bin bCourse;
    bCourse.fillBin(0, exams1.size()-1, students.size());
    // Lav trækning for hver studerende
    for (Student& s : students) {
        s.draw(bCourse);
    }
    // Udskriv resultat af lodtrækning for hver studerende
    std::cout << "Studerende og brugernavne med lodtr\x91kning" << std::endl;
    for (Student& s : students) {
        std::string examDraw = s.getExam(exams1);
        std::cout << std::left << std::setw(22) << s.getName() << " "
                  << std::setw(10) << s.getUsername() << " " << examDraw <<
std::endl;
    }


    std::cout << std::endl << "Anden lodtr\x91kning" << std::endl;
    // Anden lodtrækning
    bCourse.fillBin(0, exams2.size()-1, students.size());
    for (Student& s : students) {
        s.draw(bCourse);
    }
    for (Student& s : students) {
        std::string examDraw = s.getExam(exams2);
        std::cout << std::left << std::setw(22) << s.getName() << " "
                  << std::setw(10) << s.getUsername() << " " << examDraw <<
std::endl;
    }


    std::cout << std::endl << std::endl;
    std::cout << "--------------------" << std::endl;
    std::cout << "-- Opgave 1c test --" << std::endl;
    std::cout << "--------------------" << std::endl << std::endl;

    for (Student& s : students) {
        std::string examDraw = s.getExam(exams2);
        std::cout << s << " " << examDraw << std::endl;
    }
```

Opgave 2 (½-1 time, 15 point)

Vi skal her indlæse data fra forskellige filer (her anvendes blot en input stream fra testprogrammet), som indeholder point fra hhv. tællende aktiviteter og eksamensresultater fra forskellige fagligheder. Dette kunne f.eks. være de 3 tællende aktiviteter fra dette kursus samt to eksamensresultater (OOP og CPP), som vil være resultatet fra denne eksamen.

I denne opgave arbejder vi med de ideelle datasæt, hvor det antages, at alle studerende deltager i eksamen. Det vil altså sige, at alle der har deltaget i en tællende aktivitet også har deltaget i eksamen.

Til at løse denne opgave skal vi bruge klasserne *Grade* samt *Gradebook*. *Gradebook* er en samling af alle *Grades*, hvor den studerendes aktiviteter summeres op. Klassen *Grade* er givet på forhånd og håndterer en værdi mellem 0 og 100 samt en vægt på karakteren. Klassen *Grade* er givet herunder:

```
class Grade
{
public:
    Grade(int point = 0, float weight = 100) : mPoint(point), mWeight(weight) {}

    int getPoint() const { return mPoint; }
    float getWeight() const { return mWeight; }
    float weightedGrade() const { return mPoint*mWeight/100; }

private:
    int mPoint;     // Point fra 0 til 100%
    float mWeight;  // Vægt fra 0 til 100%
};
```

a) Gradebook

Filerne kan indeholde tomme linjer og dette skal håndteres i programmet.

Klassen Gradebook har deklarationen:

```
class Gradebook
{
public:
    Gradebook();

    void readResults(std::istream& ist);
    void printResults(std::ostream& ost) const;

private:
    std::map<std::string, std::vector<Grade> > mGrades;
};
```

Her skal alt implementeres:

- Member variablen *mGrades* er et map, som indeholder en string (brugernavn) som key og en vector til data indeholdende *Grade*-objekter. Dvs. for hvert brugernavn kan der registreres et sæt af forskellige point for de forskellige aktiviteter.
- Funktionen readResults skal læse alle data fra en input stream med dataformatet forklaret her
    - Første linje indeholder vægten af karakteren
    - Følgende linjer er i formatet ”brugernavn[mellemrum]point”
    - Der kan være blanke linjer, hvor der ikke er noget tekst

- Funktionen printResults skal formatere output pænt til output stream *ost* i følgende format
  - For hvert brugernavn (én linje)
    - For hver karakter udskrives karakteren og vægten af karakteren i parentes herefter.
    - Den totale vægtede sum beregnes til sidst og står sidst i linjen.
    - Alle tal skal være højrejusteret.
    - Kommatal vises med et ciffer efter kommaet.
  - Eksempel:

```
prhol02   100 (  2.5)  100 (  2.5)  100 (  5.0)  100 ( 45.0)  100 ( 45.0)  100.0
```

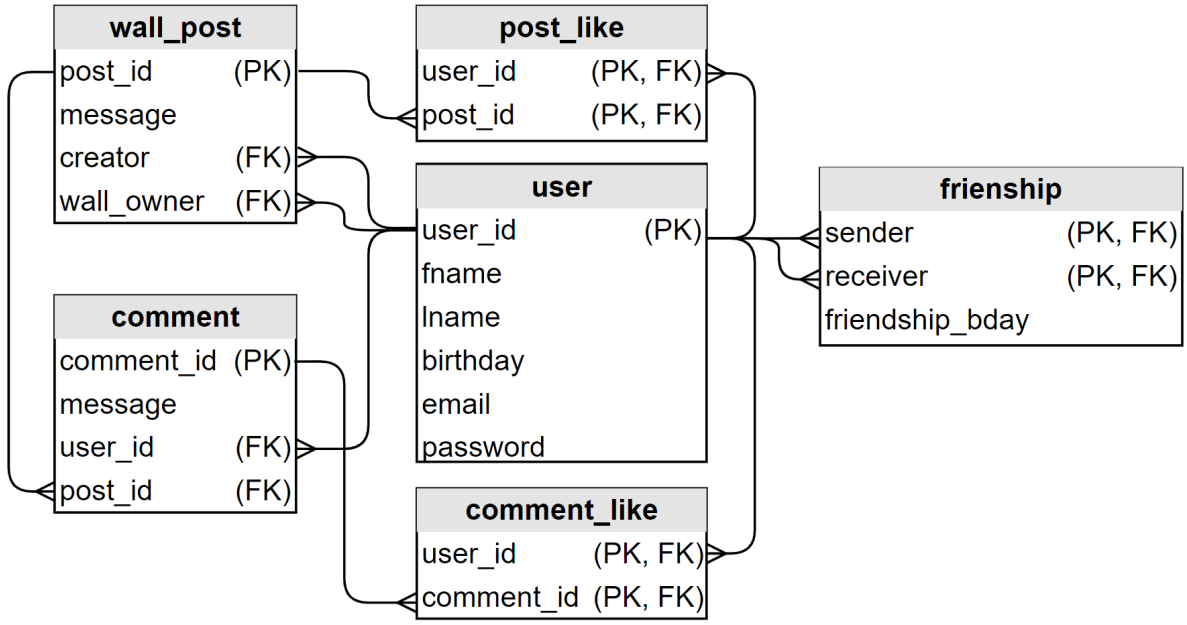
Test

Se test-kode herunder (HUSK screendump):

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

    Gradebook gBook;

    std::string activityCPP1 = "2.5\n";
    std::string activityCPP2 = "2.5\n";
    std::string activityOOP1 = "5\n";
    std::string examCPP = "45\n";
    std::string examOOP = "45\n";
    activityCPP1 += "prhol02 100\njejen20 80\nninie20  20\njajol20 50\n\n";
    activityCPP2 += "prhol02 100\njejen20 80\nninie20  50\njajol20 35\njejoe20 10\n";
    activityOOP1 += "prhol02 100\njejen20 80\n\n\njajol20 17\njejoe20 50\n";
    examCPP      += "prhol02 100\njejen20 100\nninie20  90\njajol20 80\njejoe20
70\n";
    examOOP      += "prhol02 100\njejen20 100\nninie20  80\njajol20 30\njejoe20
45\n";

    std::stringstream saCPP1(activityCPP1);
    std::stringstream saCPP2(activityCPP2);
    std::stringstream saOOP1(activityOOP1);
    std::stringstream seCPP(examCPP);
    std::stringstream seOOP(examOOP);
    gBook.readResults(saCPP1);
    gBook.readResults(saCPP2);
    gBook.readResults(saOOP1);
    gBook.readResults(seCPP);
    gBook.readResults(seOOP);

    std::stringstream ssoutput;
    gBook.printResults(ssoutput);
    std::cout << "Resultat er printet i stream" << std::endl;
    std::cout << ssoutput.str() << std::endl;
    std::cout << "Resultat er printet på skærm" << std::endl;
```

Opgave 3 (Databaser og SQL, 1 time, 25 point)

Opgaven tager udgangspunkt i nedenstående database



Der er tale om en simplificeret udgave af Facebooks database til håndtering af brugere på platformen. En bruger kan have mange venner, lave mange posts og mange kommentarer. Friendship tabellen indeholder ID på afsender og modtager af venneanmodninger. En venneanmodning er accepteret hvis venskabsfødselsdagen har en dato og ikke er NULL. Databasen kan etableres i MySQL ved hjælp af tekstfil med navnet *FacebookDDL*.

Spørgsmål 1

Find alle kommentar tekster til posts som Kim har lavet.

Spørgsmål 2

Hvem har liket Josephines posts? Find fname og lname på alle der har liket en post af Josephine

Spørgsmål 3

Du ved at én af brugerne har adgangskoden 'gosportgo' men du ved ikke hvem. Skriv et query der kan finde fname og lname for indehaveren af adgangskoden.
HINT: Kig i scriptet (*FacebookDDL*).

Spørgsmål 4

Lav en liste over navn på brugerne og tal på hvor mange kommentarer hver bruger har fået på sine posts, sorter listen efter hvem der har fået flest.

Spørgsmål 5

Du synes ikke det er så kønt med 2 fornavne. Hvis en bruger har 2 fornavne, så opdater brugeren ved at fjerne det sidste fornavn fra fname, og læg det til starten på lname,
HINT: For at finde det første ord i en streng brug REGEXP_SUBSTR(<STRING> , '^[a-z]+')

Opgave 4 (Domænemodellering, 1 time, 25 point)

En historiker/forfatter har besluttet sig for at specialisere sig i at skrive kortfattede biografier om afdøde, europæiske monarker (konger/regerende dronninger). Til det brug ønsker han/hun udviklet en it-applikation, som kan opsamle og sammenfatte relevante oplysninger om hovedpersonen.

I den første iteration af systemet skal der holdes styr på monarkens aktiviteter, fx at han/hun på en bestemt dato mødtes med et antal navngivne ministre på et bestemt sted og diskuterede et eller flere emner. Der skal tillige holdes styr på monarkens familieforhold, både hvad angår afdøde og nulevende personer.

Systemet skal som minimum kunne:

- Lave en tidslinje over monarkens aktiviteter og opholdssteder samt hvem der var til stede.
- Liste samvær med navngivne personer, fx for at opnå overblik over hvem der fortrinsvis har påvirket monarken.
- Besvare diverse forespørgsler som fx
    - Hvor længe og hvornår monarken har opholdt sig på et bestemt sted.
    - Angive opholdssted ved input af en dato.
    - Hvornår monarken har diskuteret et bestemt emne med ministre eller rådgivere.

Eksempel: systemet skal kunne registrere følgende (fiktive) aktivitet for en monark. I dette tilfælde den danske konge Frederik 3. (1609-1670).

Den 16. marts 1649 holdt kongen møde med to af sine ministre, finansminister Corfits Ulfeldt og udenrigsminister Hannibal Sehested. Ministrene var begge familiemedlemmer, idet Ulfeldt var gift med kongens halvsøster Leonora Christina og Sehested med en anden af kongens halvsøstre, Christiane. Mødet blev holdt på Københavns Slot og handlede om flådens bemanding.

Den centrale artefakt, som du skal udarbejde, er:

- Domænemodel

Og du bestemmer selv, hvilke 'hjælpeartefakter' (fx brugsmønstre, kontrakter) du vil lave. Husk at en artefakt skal kun udarbejdes, hvis den skaber værdi for systemet.

Det er vigtigt, at du argumenterer for de trufne valg i de tilfælde, hvor domænemodellen måske forekommer utilstrækkelig eller tvetydig. Den gode besvarelse indeholder således kommentarer til de væsentligste valg af de indgående elementer (klasser, attributter og relationer mellem klasser).