Opgaver re-eksamen Sommeren 2023

Dette eksamenssæt indeholder 3 opgaver

Hver opgave vægter i den samlede karakter med de %-satser, der står ved hver af dem. De enkelte delopgaver indgår med forskellig vægt i den samlede vurdering.

Der er i opgaverne angivet en række header- og implementeringsfiler (.h, eller .cpp), der skal skrives for at løse opgaverne ved at skrive den relevante kode.

Alle disse header- og implementeringsfiler skal ved afleveringen samles i én fil, der afleveres i PDF-format på Digital Eksamen. Det skal fremgå af denne PDF-fil, hvilken opgave og fil de enkelte kode-dele kommer fra. Dvs. at kodefiler ikke skal afleveres enkeltvis eller som zip fil, ligesom Visual Studio solutions og projects ikke skal afleveres. Al kode, der skal bedømmes, skal stå i denne ene fil, der vil ikke blive kigget på kodefiler, der vedlægges som bilag.

Er der spørgsmål, der kræver et tekstsvar, noteres det i koden på det passende sted, evt. som en kommentar.

Der skal ikke inkluderes screen dumps af udskrifter fra programmet i afleveringen, men det er tilladt. I så fald kan de være i samme PDF fil, eller som bilag til afleveringen. Udskrifterne fra de kørende programmer vil først og fremmest være til fordel for dig, for at tjekke, at du har løst opgaven korrekt.

Der er ingen kodebilag til dette eksamenssæt.

Husk angivelse af eksamensnummer på første side.

Opgave 1 (25 %)

Lav funktioner med følgende prototyper

```
void mid_value(const double* left, const double* right, double* middle);
void min_max_value(double left, double right, double* min, double* max);
```

mid_value har tre double pointers som input og den skal erstatte værdien som refereres af pointeren middle med værdien midt i mellem værdierne som refereres af pointere left og right
Den har denne prototype selvom den kun returnerer een værdi, for at du kan demonstrere brug af pointere til at definere input og output.
min_max_value har 2 double værdier og 2 double pointers som input, og skal erstatte værdierne som refereres af min og max med henholdsvis den mindste og den største værdi af de første. parametre

a)

Implementer funktionerne med de beskrevne prototyper og funktionalitet

b)

Implementer en main funktion til et program som beskrevet nedenfor

Funktionerne skal testes. Dette skal gøres ved, at der skrives et program, hvor brugeren bliver bedt om at indtaste to decimaltal. Der oprettes yderligere variable, og de 2 funktioner kaldes korrekt med første tal som første parameter left og andet tal som anden parameter right. Programmet udskriver derefter indholdet af de variable, der blev anvendt som nu indeholder middle, min og max.

Opgave 2 (25 %)

BMR betyder Basal Metabolic Rate og er et tal, der angiver mængden af energi ved hvile, der skal anvendes til de basale kropslige funktioner pr døgn. Beregningen involverer faktorer vægt, højde, alder og køn og ud over dette en række konstanter. Enhederne der anvendes er:
vægt:kilogram, højde:centimeter, alder:år og bmr:kcal pr døgn

Formlen for BMR er for hankøn

$$BMR = 88.362 + 13.397 * vægt + 4.799 * højde - 5.677 * alder$$

og for hunkøn

$$BMR = 447.593 + 9.247 * vægt + 3.098 * højde - 4.330 * alder$$

I denne opgave skal der skrives et program, der kan udregne BMR (Basal Metabolic Rate) for indtastede værdier af vægt i kilogram, højde i m, alder i år og køn.

a)

Bed brugeren om at indtaste m for hankøn eller f for hunkøn og gem informationen i en variabel. hvis brugeren indtaster noget forskelligt fra m eller f skal variablen for køn sættes til hunkøn og en besked om dette udskrives til brugeren

b)

Bed brugeren om at indtast vægt,højde og alder og gem disse værdier i variable. Der skal ikke laves fejlhåndtering

c)

Implementer beregningen af BMR således at dette udskrives til brugere.

Opgave 3 (50 %)

Densitet ρ for et materiale er defineret ved

$$\rho = \frac{m}{v}$$

hvor

m er materialets masse i gram [g]

v er materialets volumen i kubikcentimeter [cm^3]

ρ er det græske tegn rho, der traditionelt anvendes til at betegne densitet

Eksempler på densiteter for givne materialer er

$$\rho_{\text{gold}} = 19.30 \frac{g}{cm^3}$$
$$\rho_{\text{silver}} = 10.50 \frac{g}{cm^3}$$

a)

Definer i en h fil en klasse MineralSample, der har private attributter for vægt og volumen. Vælg selv navn og datatype for disse attributter.

b)

Argumenter for valget af data typer.

c)

Definer yderligere i h filen

- En **constructor** med parametre svarende til vægt og volumen
- **Metoderne** `is_gold()`, `is_silver()`, som skal returnerer hvorvidt en MineralSample er af guld eller ikke, og hvorvidt den er sølv eller ikke. Lad metoderne returnere den data type du mener er korrekt og argumenter for dette.

d)

Implementer klassen MineralSample i tilhørende cpp fil efter følgende specifikation

- **constructor** må ikke tillade hverken vægt eller volumen at være negative. Hvis dette er tilfældet udskrives en fejl til konsollen og vægt sættes til 0 og volumen til 1. Yderligere hvis volumen er 0 udskrives en fejl til konsollen og vægt sættes til 0 og volumen til 1.
- Metoden *is_gold()* skal implementeres ved hjælp af den opgivne densitet for guld således at den returnerer om en given instans af mineral_sample har densitet ρ der opfylder at den absolutte værdi af differensen mellem ρ og ρ_{gold} er mindre en $5*10^{-2}$. Hints: den absolutte værdi kan findes med funktionen *std::abs*, som findes i biblioteket <cmath>. $5*10^{-2}$ skrives som 5.0e-2 i C++.
- Metoden *is_silver()* skal implementeres på samme måde hvor ρ_{silver} anvendes.

e)

Lav en main funktion der tester klassen MineralSample efter specifikation

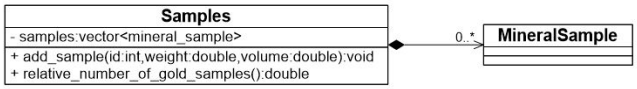
- Lav tre objekter af klassen MineralSample der tester fejl håndtering ved vægt <0, volumen <0 og volumen = 0
- Lav en test af metoderne is_gold(), is_silver() ved at definere yderligere objekter, kalde disse metoder og udskrive deres resultater.

Tabellen i figur 1 illustrerer data for tre samples.

Samples	Vægt	Volumen	is_silver	is_gold
sample 1	19.3	1	nej	ja
sample 2	105.0	10	ja	nej
sample 3	100.0	100.0	nej	nej

Figur 1: Samples UML diagram

I de følgende delopgaver skal der implementeres endnu en klasse som vi navngiver **Samples**. Denne klasse skal indeholde en liste af *MineralSamples* implementeret ved en **std::vector** og indeholdende funktionerne **add_sample** og **relative_number_of_gold_samples**. (Se figur 2)



Figur 2: Samples UML diagram

f)

Lav klasse deklarationen i en header fil samples.h

g)

Implementer klassen *Samples* **i en tilhørende cpp fil.**

- **Metoden** *add_sample* **skal oprette et** *MineralSample* **objekt og tilføje dette til vector** *samples*
- **Metoden** *relative_number_of_gold_samples* **skal returnere antallet af samples der er guld relativt til total antallet af samples som et kommatal. Hvis antallet af samples er 0 skal der returneres 0**

h)

Test klassen *Samples* **ved at tilføje kode til main funktionen, der opretter et** *Samples* **objekt og i dette objekt tilføjer en række** *MineralSample* **objekter, og dernæst kalder funktionen** *relative_number_of_gold_samples* og udskriver resultatet.