Opgaver eksamen Vinteren 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, og .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(30%)

Et system af to parallelle plader med et givet materiale i mellem pladerne udgør en elektrisk *kapacitor*. Kapacitansen afhænger af flere faktorer. Den kan udtrykkes ved følgende formel

$$C = \epsilon \frac{A}{d} \tag{1}$$

hvor

C er kapacitansen målt i enheden F.

ϵ er primitiviteten (materialets evne til at frembringe en elektrisk fluxtæthed, når det påvirkes med en elektrisk feltstyrke) af det dielektriske materiale anvendt i kapacitoren målt i enheden F/m.

A er overflade arealet af *en* plade målt i enheden m^2.

d er afstanden mellem pladerne målt i enheden m..

Der oplyses følgende data for epsilon

- `Vacuum:` $\epsilon = 8.85 \times 10^{-12}$ F/m. *Hint*: tallet skrives som `8.85e-12` i C/C++
- `Polyethelene:` $\epsilon = 2.25$ F/m
- `Mica:` $\epsilon = 7.00$ F/m
- `Keramik:` $\epsilon = 9.00$ F/m

a)

Implementer i henholdsvis en h-fil og en cpp-fil funktionen `capacitance` med følgende prototype:

```
double capacitance(double epsilon, double area,double distance);
```

som beregner kapacitansen i følge ovenstående formel (1).

b)

Implementer en `main` funktion i en fil program.cpp, der opfylder følgende kriterier (1-3)

1. Udskriv til brugeren:
   ```
   Dette program beregner kapacitans mellem to parallelle plader med et givet materiale

   Indtast materiale, areal (i enheden kvadratmeter) og afstand (i enheden meter). Materialet
   kan være: vakuum, polyethelene, mica eller keramik:
   ```

2. Indlæs de indtastede værdier i lokale variable defineret ved

   ```
   double afstand;
   double areal;
   string materiale;
   ```

3. Valider at det indtastede materiale er et af de angivne materialer. Hvis det ikke er tilfældet udskriv en fejlmeddelelse og terminer programmet. Valider yderligere at afstanden og arealet er positive tal. Hvis dette ikke er tilfældet udskriv en fejlmeddelelse og terminer programmet.

c)

Anvend den tidligere konstruerede funktion

```
double capacitance(double epsilon, double area,double distance);
```

til at beregne kapacitansen og udskrive denne til brugeren i `main` funktionen.

Opgave 2(25%)

a)

Lav en funktion med følgende prototype i en passende h-fil

```
void findMax(int* p_array, size_t size, int* result)
```

b)

Implementer funktionen i en tilhørende cpp-fil. Funktionen skal implementeres efter følgende beskrivelse:

`p_array` er en pointer til `int` som tænkes referere til første plads i det array af længde `size` funktionen kaldes med.
Arrayet må ikke være tomt(`size=0`)

`size` er længden af det array som `p_array` refererer.

`result` er en pointer til den `int` værdi, der skal opdateres med den maksimale værdi i det array der kaldes med.

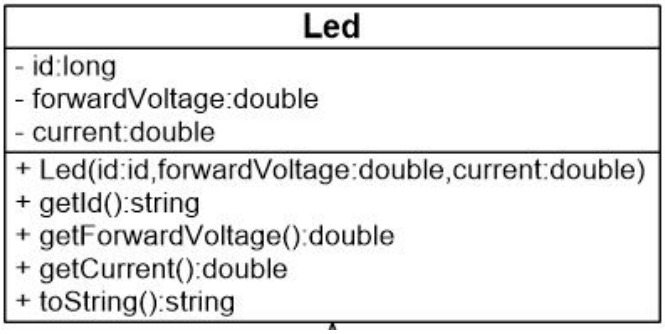
Det vil sige at funktionen skal finde det maksimale element i det array der kaldes med og tildele det til den værdi som `result` pointeren adresserer.

c)

Test denne funktion i en main funktion. Argumenter for dit valg af testcase(s).

Opgave 3(45%)

Givet følgende UML diagram som repræsenterer data for en Light Emmiting Diode(LED). Løs følgende opgaver



Figur 1: Class Diagram Led

a)

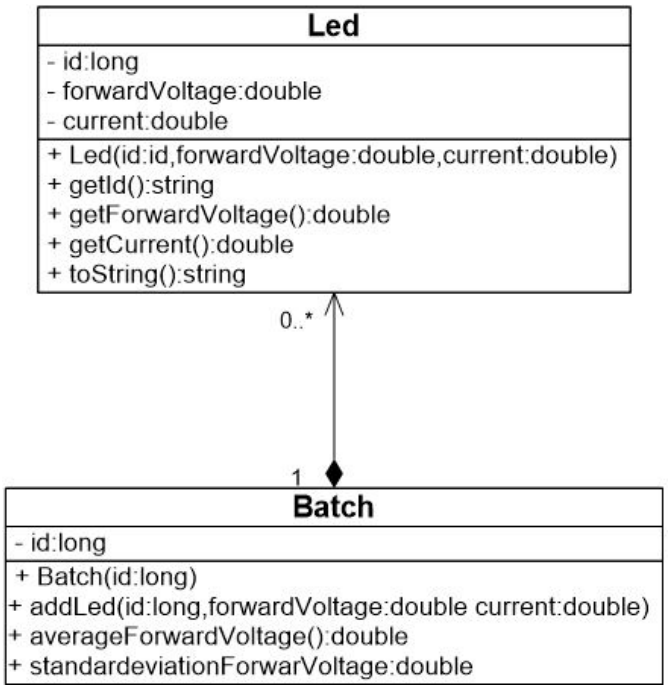
Implementer klasse prototypen i en h-fil

b)

Implementer klassens metoder i en tilhørende cpp-fil. Get-metoderne returnerer respektive attributter og `toString` metoden returnerer en streng repræsentation af alle `Led` objektets attributter.

c)

Betragt nu følgende UML diagram Her ses at `Batch` står i en 0 til mange relation til klassen `Led` og at relationstypen



Figur 2: Class Batch and Led

er en komposition.

Udover at Batch skal implementere kompositionen, indeholde en attribut id af typen `long` og have den specificerede constructor, indeholder den to metoder

- `averageForwardVoltage()`: Denne metode returnerer middelværdien(`average`) af alle tilknyttede Led objekters `forward voltage`.
- `standardDeviationForwardVoltage()`: Denne metode returnerer spredningen(`standard deviation`) af alle tilknyttede `Led` objekters `forward voltage`.

Formlen for middelværdi(`average`) er:

$$average = (fv_1 + fv_2 + ... + fv_n)/n \tag{2}$$

hvor fv_i er den enkelte Led objekters `forwardVoltage` og n er antallet af `Led` objekter i et givet `Batch` objekt.

Formlen for spredning(`standard deviation`) er:

$$standard\ deviation = \sqrt{((fv_1 - average)^2 + (fv_2 - average)^2 + ... + (fv_n - average)^2)/(n-1)} \tag{3}$$

hvor fv_i er det enkelte `Led` objekts `forwardVoltage` og $average$ er den beregnede middelværdi for det givne `Batch` objekt.

d)

Implementer i en h-fil deklarationen for klassen `Batch`.

e)

Implementer i en tilhørende cpp-fil constructor og specificerede metoder.

f)

Test Batch klassens metoder i en main fil. Argumenter for dit valg af testcase(s)