AARHUS UNIVERSITET
INGENIØRHØJSKOLEN

EKSAMEN

Kursus:	I1OPRG/ E1OPRG – Objektbaseret programmering
Eksamensdato:	11. juni 2019, kl. 09.30-11.30 (FP 09.30 – 12.15)
Eksamenstermin:	Sommereksamen 2019
Ingeniørhøjskolen udleverer: 4 stk. kladdepapir	
Praktiske informationer: **Digital eksamen** Opgaven tilgås og afleveres gennem den digitale eksamensportal. Eksamensbesvarelsen kan kun afleveres digitalt. Det er ikke muligt, at digitalisere håndskrevne bilag. Eksamensbesvarelsen skal afleveres i PDF-format. Husk angivelse af navn og studienummer på alle sider (første side er nok, se nedenfor), samt i dokumenttitel/filnavn. Husk at uploade og aflevere i Digital eksamen. Du vil modtage en elektronisk afleveringskvittering, straks du har afleveret. Husk at aflevere til tiden, da der ellers skal indsendes dispensationsansøgning.	
Hjælpemidler: Alle hjælpemidler må benyttes, herunder internettet som opslagsværktøj, men det er IKKE tilladt at kommunikere med andre digitalt.	
Særlige bemærkninger: Der er en eller flere filer som bilag, der kan fines på Digital Eksamen, sammen med eksamenssættet.	
Ansvarlig underviser: Frank Bodholdt Jakobsen	

Instruktion

Dette eksamenssæt indeholder 3 opgaver, der enten skal løses i C eller C++, som angivet for hver enkelt opgave.

Hver opgave vægter i den samlede karakter med de %-satser, der står ved hver af dem.

Der er i opgaverne angivet en række header- og implementeringsfiler (.c, .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 i denne PDF-fil, hvilken opgave og fil de enkelte kode-dele kommer fra. Dvs. at kode-filer **ikke** skal afleveres enkeltvis eller som zip fil, ligesom Visual Studio solutions og projects **ikke** skal afleveres.

Husk angivelse af navn og studienummer på første side.

Der er en eller flere filer med som bilag, som kan findes på Digital Eksamen, samme sted som eksamenssættet.

Opgave 1 (40 %)

Denne opgave skal løses i C.

Opgaven har som mål at lave et program, der kan bruges til at spille "Blackjack", hvor det gælder om at få flere point end banken, uden at overstige 21 point. Spilleren trækker kort indtil spilleren siger stop eller summen af pointværdien for spillerens kort overstiger 21, hvorved spilleren har tabt med det samme.

Kortene har en pointværdi fra og med 1 (et) til og med 11 (elleve).

Første skridt er at simulere at der trækkes et tilfældigt kort ved at trække et tilfældigt tal.

a) **Skriv prototypen til funktionen `traekEtKort()`, der ikke har nogen parametre, og returnerer et heltal, som den skal stå i header-filen Kort.h.**

Du skal nu implementere denne funktion, den skal returnere et tilfældigt tal fra og med 1 til og med 11.

b) **Skriv implementationen af `traekEtKort()`, som den skal stå i implementationsfilen Kort.c. Du skal bruge funktionen `rand()` til at finde det tilfældig tal.**

Du skal nu teste denne funktion, ved at skrive et program (`main()`), der kalder `traekEtKort()`.

c) **Skriv hele filen Opgave1C.c, med en `main()` funktion til et program, der tester `traekEtKort()` ved at kalde den 10 gange og udskrive returværdien**

Opgaven fortsættes på næste side!

Du skal nu skrive noget kode, der gennemfører spillerens del af spillet, ved at gøre det muligt for brugeren at trække mindst et kort, og blive ved at kunne trække, indtil brugeren stopper eller summen af kortenes point bliver større end 21.

Følgende pseudokode kan bruges til at strukturere programmet:

Nulstil summen
Udfør

Træk et kort
Udskriv pointværdien
Læg pointværdien til summen
Udskriv summen

***Hvis** summen er mindre end eller lig med 21*
Spørg brugeren, om der skal trækkes et kort til
Indlæs svaret og gem det
Ellers
Udskriv besked "Du har tabt"

***Så længe** summen er mindre end eller lig 21 **og** brugerens svar er, at der skal trækkes et kort til*

d) Skriv hele filen Opgave1D.c, med en `main()` funktion til et program, der gennemfører spillet som beskrevet.

Opgavesættet fortsættes på næste side!

Opgave 2 (20 %)

Denne opgave kan valgfrit løses i C eller C++ (der skal ikke bruges klasser).

I denne opgave skal der arbejdes med en funktion, der kan ombytte indholdet i 2 heltals-variable.

Man kan ombytte to heltal `a` og `b` således:

```
int temp;

temp = a;
a = b;
b = temp;
```

Hvis `a=5` og `b=6` *før* kørslen af ovenstående kode, vil `a=6` og `b=5` *efter* kørslen af ovenstående kode (prøv selv efter).

Nu skal du skrive funktionen `ombyt()`, som tager 2 variable af typen `int` som parametre, der skal overføres til funktionen ved call-by-reference. Funktionen skal sørge for, at indholdet af to variable byttes om.

Prototypen til funktionen er som angivet nedenfor. Den skal stå i header-filen Ombyt.h, som du selv skal skrive:

```
void ombyt(int *i1, int *i2);
```

a) **Skriv implementationen for funktionen `ombyt()`, som den skal stå i implementationsfilen Ombyt.cpp**

Nu skal funktionen testes på to heltals-variable, der initialiseres med forskellige værdier. Deres indhold udskrives før og efter at `ombyt()` er blevet kaldt med de to variable som parametre.

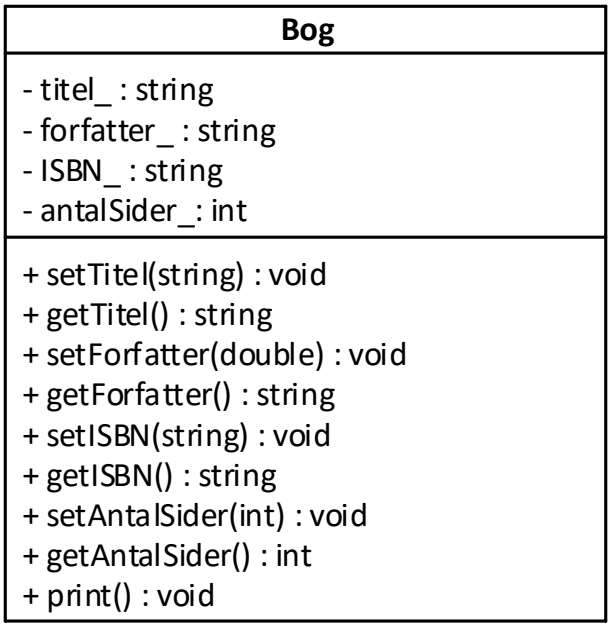
b) **Skriv hele filen Opgave2.c eller Opgave2.cpp, med en `main()` funktion til et program, der tester `ombyt()` som beskrevet.**

Opgavesættet fortsættes på næste side!

Opgave 3 (40 %)

Denne opgave skal løses i C++.

Denne opgave drejer sig om en klasse Bog, der kan antages at være en del af et bibliotekssystem. UML klassediagrammet for klassen er angivet nedenfor:



Om klassen Bog gælder følgende:

Attributterne skal opfylde følgende betingelser:

Navn	Beskrivelse	Gyldige værdier	Default værdi
titel_	Bogens titel	Titlen må ikke være tom, dvs. det skal være forskelligt fra den tomme streng "".	"Ukendt"
forfatter_	Bogens forfatter	Forfatteren må ikke være tom, dvs. det skal være forskelligt fra den tomme streng "".	"Ukendt"
ISBN_	Bogens ISBN nummer, som string, ikke som tal	ISBN skal være nøjagtig 13 tegn langt, og alle tegn skal være et af cifrene fra '0' til '9'	"1234567890123"
antalSider_	Antallet af sider i bogen	Antallet af sider skal være større end 0	1

Opgaven fortsættes på næste side!

Klassen `Bog` har følgende kombinerede default/parametriserede constructor med default argumenter.

```
Bog(string titel = "Ukendt",
      string forfatter = "Ukendt",
      string ISBN = "1234567890123",
      int sider = 1);
```

Parametre:

`titel`: bogens titel
`forfatter`: bogens forfatter
`ISBN`: bogens ISBN nummer
`sider`: antal sider i bogen

Beskrivelse:

Constructoren validerer parametrene og sætter de relevante attributter. Hvis en parameter ikke er gyldig, indsættes defaultværdien.

Om metoderne for klassen gælder:

```
void print() const;
```

Parametre:

Ingen

Beskrivelse:

Metoden udskriver bogens informationer fra attributterne i dette pæne format:

```
Martin Andersen Nexø: Ditte Menneskebarn
ISBN: 9788702287660
728 sider
```

`Set`-metoderne er almindelige metoder, der sætter de tilsvarende attributter med validerede værdier fra parametre ud fra ovenstående beskrivelse, og indsætter default værdier, hvis parametrene ikke indeholder gyldige værdier.

`Get`-metoderne er almindelige metoder, der bare returnerer den tilsvarende attribut.

Opgaven fortsættes på næste side!

a) Skriv hele header-filen Bog.h for klassen `Bog`**.**

Samtlige klassens metoder, undtagen `print()` og `setISBN(string)` er **allerede** implementeret i filen **Bog.cpp** som kan findes blandt bilagene. Du skal bruge denne fil til at løse de følgende opgaver.

b) Implementer metoderne `print()` og `setISBN(string)` i filen Bog.cpp.

c) Implementer constructoren Bog i filen Bog.cpp ud fra specifikationen ovenfor.

Nu skal klassen testes i et program, ved at skrive en `main()` funktion i en ny fil, Opgave3D.cpp.

Programmet skal oprette et `Bog`-objekt med passende parametre og herefter udskrive objektets indhold.

d) Test klassen `Bog` **ved at skrive hele filen Opgave3.cpp med** `main()` **funktionen til et program, der udfører ovennævnte test.**

Som en del af et biblioteksystem kunne man forestille sig et kartotek med bøger. Dette kan oprettes som et array af `Bog` objekter.

Dette skal testes ved at skrive et program, der først opretter et array med 4 `Bog`-objekter med forskelligt indhold, derefter gennemløber arrayet med en `for` løkke og udskriver hver enkelt `Bog`-objekt.

e) Simulér et kartotek ved at tilføje kode til testprogrammet fra opgave d), der opfører sig som beskrevet ovenfor.