TDIU20 – Tentaregler

Inloggning

Logga in i tentasystemet genom att välja session ”exam system” och logga in med ditt vanliga LiU-ID. Välj inte att ha denna session som standardsession. Verifiera att dina uppgifter stämmer och förbered din tentaplats. Som vanligt är det inte tillåtet att ha väskor eller jackor vid sin skrivplats och mobiltelefoner ska ligga avstängda i jacka eller väska. Ta fram ditt LiU-kort och invänta tentavakt för att få ett engångslösenord.

Hjälpmedel

Följande får tas med på tentan:

- En bok om c++. För boken gäller följande regler:
  - Kommentarer/noteringar som direkt rör text och exempel på sidan i fråga får finnas i sidmarginalen.
  - Egna sidflikar för att enkelt kunna hitta t.ex. de olika kapitlen är tillåtna.
  - Inga extra ark eller lappar, lösa eller fastsatta, får finnas.
  - Tomma sidor, insidan av pärmarna, försättsblad, etc., får inte innehålla programkod.
- Maximalt en sida med egna, handskrivna anteckningar
- Penna för att anteckna under tentan. Ni kommer förses med blanka papper

Följande får INTE tas med:

- Elektroniska hjälpmedel såsom miniräknare och mobiltelefon

Utloggning

När du är nöjd med ditt betyg (som står i tentaklienten) kan du avsluta och logga ut som vanligt i menyn. Klicka sedan på ok följt av knappen avsluta tentan. Observera att du när du gjort detta inte kan logga in igen. Lämna inte din plats innan vanliga inloggningsskärmen syns.

Tentaregler

Tentan består av fyra uppgifter klassificierade som grundnivå eller avancerad. För godkänt betyg på tentan krävs lösning av en uppgift på grundnivå. För betyg 4 krävs dessutom lösning av en avancerad uppgift och för betyg 5 krävs en grunduppgift samt båda avancerade uppgifterna.
För att en uppgift ska anses godkänd krävs följande:

- att man noga följt alla instruktioner och krav ställda i uppgiften
- din kod följer god programmeringsstil (se labseriens rättningsguide)
- att klasser har ett tydligt ansvar och funktioner har en väl definierad uppgift
- att din kod har bra inkapsling och resurshantering

Bonus från labserien

Om du har bonus från labserien minskas antalet avancerade uppgifter som krävs med ett, dvs för betyg 4 krävs endast en grunduppgift och för betyg 5 krävs en av varje. Bonusen är endast giltig det år den erhölls.

Frågor om uppgifter

Frågor om tentan i stort eller uppgiftspecifika frågor ska ställas via tenta-klienten. Detta för att vi ska ha en historik av konversationen samt för att vi ska kunna ge samma hjälp till olika studenter.

Systemfrågor

Om du har systemfrågor som t.ex. problem med tentaklienten eller terminalen räcker du upp handen så kommer en assistent och hjälper till.

Uppgift 1 - Grundnivå

Skriv din lösning på en fil med namnet `uppgift1.cc`

I denna uppgift ska du skapa klasser som beskriver en hand i kortspelet Blackjack. En hand består av minst två kort. Ett kort har en av vanliga färgerna hjärter, ruter, klöver, spader och ett värde 2-10, knekt, dam, kung eller ess. Speciellt i Blackjack är att knekt, dam och kung värderas 10 och ess antingen 1 eller 11 beroende på vad som är mest fördelaktigt för spelaren. Målet är att korten i handen ska ge högsta möjliga summa utan att gå över 21.

Skapa en klass `Card` för att representera ett vanligt spelkort. Denna klass ska lagra både färg och värde för ett kort. Skapa därefter en klass `Hand` som innehåller en `vector` av `Card`. Denna klass ska ha en medlemsfunktion för att skriva ut handen samt en funktion för att beräkna värdet av handen genom att summera kortens värden i enlighet med ovanstående beskrivning. Ett tips är att beräkna ett ess som värde ett och om totalsumman är 11 eller mindre och du har minst ett ess lägger du till 10 till summan.

Ditt program ska skapa tre separata händer innehållande de kort som står i körexemplet nedan. Du ska alltså hårdkoda in de tre separata händerna och för varje hand skriva ut korten som handen består av och handens värde i spelet Blackjack.

Körexempel 1

```
Handens kort:
    spader ess
    hjärter 7
Värde: 18

Handens kort:
    klöver knekt
    ruter dam
    ruter 7
Värde: 27

Handens kort:
    ruter ess
    hjärter 9
    spader ess
Värde: 21
```

OBS: Uppgiften går ut på att skapa klasserna enligt beskrivningen. Inte att endast få samma utskrift som körexemplet.

Uppgift 2 - Grundnivå

Skriv din kod på en filer med namnen `uppgift2.cc, book.h` och `book.cc`

Skapa en klass som representerar en bok. En bok består av ett antal sidor, är skriven av en författare och har en titel.

En bok ska vara möjlig att jämföra mot en annan bok med den vanliga operatorn <, och gå att läsa in med vanlig operator för formaterad inläsning **>>** samt skriva ut med **<<**. Jämförelsen (bokstavsordning) sker i första hand på författarens namn och därefter titel.

Huvudprogrammet ska läsa in tre böcker och därpå skriva ut dem i sorterad ordning, minst först. Sorteringen kan exempelvis lösas med tre jämförelser.

En bok matas in och skrivs ut enligt givna körexempel.

Körexempel 1 (Användarinmatning är i fet stil)

```
Mata in bok 1: Nu ska karusellen gå; Andersson; 1
Mata in bok 2: Programming C++; Stroustrup; 582
Mata in bok 3: UML Distilled; Fowler; 78

I biblioteket ska böckerna stå i följande ordning:
Andersson; Nu ska karusellen gå; 1
Fowler; UML Distilled; 78
Stroustrup; Programming C++; 582
```

Körexempel 2 (Användarinmatning är i fet stil)

```
Mata in bok 1: Problem Solving with C++; Savitch; 340
Mata in bok 2: Programming C++; Stroustrup; 582
Mata in bok 3: Absolute C++; Savitch; 372

I biblioteket ska böckerna stå i följande ordning:
Savitch; Absolute C++; 372
Savitch; Problem Solving with C++; 340
Stroustrup; Programming C++; 582
```

TIPS: Använd getline med semikolon som radslutavgränsare för att läsa in titel och författare.

Uppgift 3 - Avancerad

Kopiera filen `given_files/uppgift3.cc` till din hemmapp och modifiera denna.

Skapa en klass, med namn `Copy_Counter`, som håller reda på hur många gånger ett visst `Copy_Counter`-objekt har blivit kopierat. När den sista kopian av ett objekt försvinner ska antal kopieringar som gjorts skrivas ut.

Se det givna programmet för exempelutskrift och förklaring.

TIPS 1: Ett objekt lagrar något som gör att både objektet och en (grund) kopia kan komma åt en räknare. Här behöver alla kopior av ett objekt dela på samma räknare för att kunna veta både hur många kopieringar som totalt gjorts samt hur många objekt som faktiskt finns kvar. Detta görs lättast med pekare.

TIPS 2: Tänk noga igenom vilka medlemsfunktioner som anropas i de olika stegen i den givna koden för att reda ut hur denna klass ska fungera.

Uppgift 4 - Avancerad

Kopiera filen `given_files/uppgift4.cc` till din hemkatalog och modifiera denna. Filen innehåller ett givet huvudprogram som inte får ändras.

I denna uppgift ska du skapa en polymorfisk klasshierarki för att representera olika dryckesbehållare. Alla dryckesbehållare har en beskrivning och kan ibland vara stapelbara. Det finns två kategorier av dryckesbehållare, de som är designade för varma drycker och de som är designade för kalla drycker. De som är designade för varma drycker kan vara isolerade (exempelvis en termos).

Designa din klasshierarki enligt följande beskrivning:

- Skapa en basklass med namn `Container` med datamedlemmar `std::string description` och `bool stackable`. Det ska inte gå att skapa objekt av denna typ.
- Skapa en direkt subklass till `Container` med namn `Hot_Container` för att representera dryckesbehållare för varma drycker. När en `Hot_Container` skapas ska den ta emot information om beskrivningen samt om den är stapelbar. Dessutom ska den lagra om den har ett öra eller inte. Även detta ska kunna anges när en `Hot_Container` skapas. Om inget anges ska det antas att den har ett öra.
- Skapa en subklass till `Hot_Container` med namn `Thermos`. En termos har aldrig ett öra men har ingen extra information jämfört med `Hot_Container`.
- Skapa en annan direkt subklass till `Container` med namn `Cold_Container`. Förutom beskrivning och stapelbarhet ska `Cold_Container` även initieras med dess material (givet som en `string`, exempelvis “glas” eller “plast”).

Förutom de speciella medlemsfunktioner som krävs ska samtliga klasser i hierarkin ha följande medlemsfunktion:

- `void print()` ska skriva ut en beskrivning av objektet. Samtliga ska först skriva ut beskrivningen från `Container` och om den är stapelbar ska strängen `"[stapelbar]"` skrivas ut efteråt. Därefter skrivs olika saker ut beroende på typ av objekt enligt nedan:
  - För `Hot_Container`: Om den har ett öra ska strängen `"med öra"` skrivas ut.
  - För `Thermos`: Strängen `"(termos)"` skrivs ut
  - För `Cold_Container`: Materialet skrivs ut inom parenteser.

  Se kommentarer i den givna koden för exempel på hur utskrifterna kan se ut.

Kopiering och flytt av objekt i denna hierarki ska inte vara tillåtet på något sätt.