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.
- Ett A4-ark med valfritt innehåll på vardera sidor (går ej att ersätta med två enkelsidiga ark).
- 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.

C++ referens

Det finns tillgång till valda delar av `cppreference.com`. Du måste starta webbläsaren via menyn för att komma åt sidan.

Alias för kompilering

Det finns tre alias att använda sig av för kompilering med c++17:
`g++17` Kompilering utan varningar.
`w++17` Rekommenderas!
`e++17` Kompilering med alla varningar som fel.

Uppgift 1 - Grundnivå

Kopiera filen `given_files/{uppgift1.cc,text.h}` till din hemmapp. Filerna innehåller given kod som inte får ändras. Skriv din lösning på filerna med namnen `ebook.h,ebook.cc`.

Du ska skapa en klass för att representera en E-bok. Ett objekt av typen E-bok skapas genom att ange en sträng (ska krävas) med textinnehållet, en sidbredd (antal tecken per rad) och en sidhöjd (antal rader per sida). Som standard om inget anges ska det vara 80 tecken per rad och 40 rader per sida.

För att kunna stega igenom bokens sidor från början till slut ska det finnas en funktion som returnerar sant om det inte finns några fler sidor att visa, och en funktion som returnerar alla ord på nästa sida som en sträng.

En fördel med en e-boksläsare är att man enkelt kan ändra storlek på tecknen. Det för med sig att det ryms fler eller färre tecken och rader på sidan. I din klass ska det finnas en medlemsfunktion man när som helst kan anropa för att ändra sidbredd och sidhöjd. Om en ändring sker ska den påverka hur nästa sida visas.

En sida presenteras med ojämn högerkant (“ragged-right”). Orden ska alltså inte avstavas, utan radbyten ska ske mellan ord. Nyradstecken i bokens text är styckebrytning och ska visas med en tomrad.

TIPS: En möjlig algoritm för att få fram nästa sida korrekt:

1. Antag att du har en strängström för utskrift kallad `oss`.
2. Hämta ut en radlängd från boken.
3. Sök baklänges (`string::rfind`) efter närmaste mellanslag eller nyrad.
4. Hittades en nyrad skriver vi ut raden så långt till `oss` samt en tomrad och stegar fram motsvarande längd i textmassan.
5. Annars, om ett mellanslag hittades, skriver vi ut raden så långt och stegar fram motsvarande längd i textmassan.
6. Fortsätt tills alla rader på sidan skrivits ut.
7. Returnera innehållet i `oss` som en sträng.

Uppgift 2 - Grundnivå

Kopiera filen `given_files/{uppgift2.cc,catch.hpp}` till din hemmapp. Filerna innehåller given kod som inte får ändras. Skriv din lösning på filerna med namnen `integer.h,integer.cc`.

Skapa en klass som representerar ett vanligt heltal. Din klass ska klara både positiva och negativa tal som är dubbelt så stora tal som den datatyp du använder för att lagra talets värde. Detta åstadkoms genom att lagra talets tecken separat och talets värde som en `unsigned int`.

Ett objekt av typen ska kunna skapas genom att ange ett vanligt heltal (`int`). Som standard (om inget anges) ska heltalet få värdet 0. Noll ska dessutom alltid räknas som positivt.

Objekt av din heltalsklasstyp ska kunna matas ut på valfri utström (`std::ostream&`) och kunna jämföras. Endast jämförelserna för "likhet" och "mindre än" behöver implementeras. Ett tal ska kunna räknas upp med prefix-operatorn `++`.

Det ska även gå att konvertera dina heltal till vanliga heltal. Detta ska endast kunna ske på begäran, inte automatiskt. Om det visar sig att ditt heltal inte ryms i ett vanligt heltal ska ett undantag av typen `std::range_error` från `<stdexcept>` kastas. Största värdet för `int` går att få fram med hjälp av `numeric_limits<int>::max()` i `<numerics>`, se cppreference.

Alla testfall i det givna testprogrammet måste fungera.

Uppgift 3 - Avancerad

Kopiera filerna `given_files/{uppgift3.cc,poolparty.cc,poolparty.h}` till din hemkatalog. Filerna innehåller given kod som inte får ändras. Skriv din lösning på filerna med namnen `guest.h,guest.cc`.

I denna uppgift ska du skapa en polymorfisk klasshierarki för att representera olika typer av gäster till ett pool-party. En `Guest` har ett namn och ska kunna lämna svar på inbjudan till partyt genom att gästens medlemsfunktion `update` anropas. Update-funktionen tar emot en beskrivning av partyt som ett objekt av typen `Pool_Party` (se given kod). Svaret lämnas enbart genom en utskrift på skärmen. Det finns två huvudtyper av gäster, en `Committed_Guest` och `Hesitant_Guest`.

En "committed" gäst svarar alltid med sitt namn och "I'm delighted! Of course I'll come :-D".

En "hesitant" gäst kan vara tveksam av lite olika anledning och kan komma att avböja inbjudan. Medlemsfunktionen `will_decline` tar emot ett objekt av typen `Pool_Party` och returnerar sant om gästen avböjer. Medlemsfunktionen `get_reason` returnerar anledningen. Gästen svarar på inbjudan med sitt namn och meddelandet "Ohh! Nice, I can make it :-)" eller med sitt namn och meddelandet "Sorry," följt av den specifika anledningen.

Du ska skapa två typer av tveksamma gäster. En `Prude_Guest` är lite blyg och dess `will_decline` kommer att avböja om det finns risk att andra gäster badar nakna. I det fallet anges anledningen (från `get_reason`) "I have a dentist appointment at that time".

En `Busy_Guest` är upptagen fram till en viss timme. Timmen såväl som namnet anges när klassobjektet skapas. Gästens `will_decline` kontrollerar om partyt börjar sent nog. Om gästen avböjer svarar den med meddelandet "I can not make it until" följt av den timme hen kan anlända till partyt.

Körexempel 1

```
POOL PARTY at 18
Beatrice: Sorry, I can not make it until 19!
Patrik: Sorry, I have a dentist appointment at that time.
Caspar: I'm delighted! Of course I'll come :-D

POOL PARTY at 20 (bring bath suit to get in)
Beatrice: Ohh! Nice, I can make it :-)
Patrik: Ohh! Nice, I can make it :-)
Caspar: I'm delighted! Of course I'll come :-D
```

OBS: Uppgiften går ut på att skapa klasserna enligt beskrivning med god objektorienterad klassdesign. Inte att endast få samma utskrift som körexemplet.

Uppgift 4 - Avancerad

Kopiera filerna `given_files/{uppgift4.cc,stack.h,stack.cc}` till din hemkatalog. Skriv din lösning på filerna med namnen `stack.h,stack.cc`. `uppgift4.cc` innehåller huvudprogrammet som ej får ändras.

I de givna filerna finns en implementation av en stack. Tyvärr är koden inte helt komplett. Din uppgift är att komplettera koden med kopieringskonstruktor, flyttkonstruktor, tilldelningsoperator, flyttoperator och destruktor. Du behöver även lägga till undantagsklassen `Empty_Stack` som ska vara baserad på `std::out_of_range`.

Det givna huvudprogrammet får ej ändras och ska fungera så effektivt och säkert som möjligt. Inga minnesläckor får förekomma.

Körexempel 1

```
Mata in heltal, avsluta med Ctrl+D: 1 2 3 4 5
Mata in heltal, avsluta med Ctrl+D: 6 7 8 9 0
Du matade sedan in talen (här i omvänd ordning):
5 4 3 2 1
Du matade sedan in talen (här i omvänd ordning):
0 9 8 7 6
```