TDP004 - Tentamen

2017-08-21

Regler

- All kod som skickas in för rättning ska kompilera och vara väl testad.
- Inga elektroniska hjälpmedel får medtas. Mobiltelefon ska vara avstängd och ligga i jacka eller väska.
- Inga ytterkläder eller väskor vid skrivplatsen.
- Student får lämna salen tidigast en timme efter tentamens start.
- Vid toalettbesök ska pauslista utanför salen fyllas i.
- All form av kontakt mellan studenter under tentamens gång är strängt förbjuden.
- Böcker och anteckningssidor kan komma att granskas av assistent, vakt eller examinator under tentamens gång.
- Frågor om specifika uppgifter eller om tentamen i stort ska ställas via tentasystemets kommunikationsklient.
- Systemfrågor kan ställas till assistent i sal genom att räcka upp handen.
- Endast uppgifter inskickade före tentamenstidens slut rättas.
- Ingen uppgift kan kompletteras under tentamens sista kvart.
- En uppgift kan som regel kompletteras tills den är antingen "Godkänd" eller "Underkänd". En uppgift bedöms som "Underkänd" om ingen markant förbättring skett sedan tidigare inlämning.
- Kompilerande kod, fullständig kravuppfyllnad och följande av god stil och goda konventioner enligt god programmeringssed är krav för att en uppgift ska bedömas "Godkänd".

Hjälpmedel	En C++-bok (t.ex. Stroustrup) Ett A4-ark med egna anteckningar

Information

Betygssättning vid tentamen

Tentamen består av ett antal uppgifter på varierande nivå. Uppgifter som uppfyller specifikationen samt följer god sed och konventioner ges omdömet "Godkänd". Annars ges omdömet "Kompletteras" eller "Underkänd". Tentamen kräver två godkända uppgifter för betyg 3. Alla betygsgränser ses i tabell 1 och 2. *För betyg 3 har du alltid hela tentamenstiden, varken mer eller mindre.* (För student som från LiU fått rätt till förlängd skrivtid förlängs betygsgränserna i proportion till den förlängda skrivtiden.)

Tid	**Lösta uppgifter**	**Betyg**	**Tillgodo**
4 h	3	5	inget
2.5 h	2	4	inget
4 h	2	3	inget
4 timmar	1	|	löst uppgift

Tabell 1: Betygsättning vid förtentamen (dugga), 4 uppgifter ges

Tid	**Lösta uppgifter**	**Betyg**
3 h $+B$	3	5
4 h $+B$	4	5
4 h $+B$	3	4
2 h $+B$	2	4
5 timmar	2	3

Tabell 2: Betygsättning vid sluttentamen, 5 uppgifter ges

Bonustid ($+B$)

Ingen bonustid ges vid omtentamen.

Tillgodoräknanden

Tillgodoräknanden gäller endast under den första ordinarie tentamen i samband med kursen. Om du på förtentamen endast lyckas lösa en uppgift kan du tillgodoräkna motsvarande uppgift på sluttentamen (se tabell 1). Du har då bara en uppgift kvar till betyg 3. För högre betyg (om du löser mer än en uppgift på förtentamen) kan du inte tillgodoräkna något. Om du på sluttentamen löser en andra uppgift snabbt och vill sikta på högre betyg kan du lösa den tillgodoräknade uppgiften "igen", men den räknas endast mot högre betyg, **inte** som andrauppgift för betyg 3.

Inloggning

Innan du loggar in ska du välja "Exam system" från sessionsmenyn i nedre vänstra hörnet av inloggningsrutan. Därpå loggar du in med dina LiU inloggningsuppgifter. Du kommer nu in i tentainloggningen och ska börja med att välja språk. Ta den flagga som ser minst Engelsk ut så blir det Svenska (valet spelar ingen roll för dig som kan båda språken). Följ instruktionerna på skärmen så långt det går tills du ska mata in ett engångslösenord. Tag fram ditt LiU-kort och visa det för assistent eller vakt i sal för att få detta lösenord.

Skrivbordsmiljön

När du kommit in i tentasystemet har du en normal skrivbordsmiljö (Mate-session i Linux Mint) med grön skrivbordsbakgrund. Efter en stund kommer även din kommunikationsklient att dyka upp automatiskt. Startmenyn är nedskalad till enbart det som examinator bedömt relevant. Andra program kan fortfarande finnas tillgängliga genom att starta dem från en terminal. Observera att en del program som använder nätverkstjänster inte fungerar normalt, eftersom nätverket inte är åtkomligt.

När du är inloggad är det viktigt att du har tentaklienten igång hela tiden. Om den inte dykt upp fem minuter efter inloggning och inte heller när du startar den manuellt från menyn (fisken) tar du kontakt med assistent eller vakt i sal.

Terminalkommandon

`e++17` används för att kompilera med "alla" varningar *som fel*.
`w++17` används för att kompilera med "alla" varningar. **Rekommenderas.**
`g++17` används för att kompilera `utan` varningar.
`valgrind --tool=memcheck` används för att leta minnesläckor.

C++ referenssidor

På tentan har du *experimentiell* tillgång till referenssidorna på `http://www.cppreference.com/` via webbläsaren Chrome. Starta `chromium-browser` i terminalen eller välj lämpligt alternativ från startmenyn. Observera att allt utom referenssidorna är avstängt. Om du inte kan komma åt en sida du tycker hör till referenssidorna (som kanske blockerats av misstag) kan du skicka ett meddelande via tentaklienten. Då lösningen är på experimentell nivå kan det hända att problem uppstår, t.ex. att proxyn inte går att nå. Tag då hjälp av assistent i sal.

Givna filer

Eventuella givna filer finns i katalogen `given_files`. Denna underkatalog är skrivskyddad, så det är ingen risk du råkar ändra på dessa filer. Skrivskyddet gör dock att du måste kopiera in de givna filer du vill använda till tentakontots hemkatalog. Hur du listar och kopierar filer ska du kunna. Hemkatalogen står du i från början, och du kommer alltid tillbaka till den genom att bara exekvera kommandot `cd` i terminalen. Hemkatalogen heter alltid `/home/student_tilde` om du undrar.

Avslutning

När dina uppgiftsbetyg och ditt slutbetyg i kommunikationsklienten stämmer överens med det du förväntar och du är nöjd, eller när tentamenstiden är slut, är det dags att logga ut. Hinner du inte se ditt betyg får du höra av dig till examinator via epost efter tentamen.

Avsluta alla öppna program och tryck på knappen märkt "Exit" i menyn längst ner på skärmen och välj "ok". Vänta ett tag och tryck sedan på knappen "Avsluta tentamen" när det är möjligt. När detta är gjort är det omöjligt att logga in igen. Lämna inte datorn förrän du ser den vanliga inloggningsskärmen med blå bakgrund. Anmäl till assistent eller vakt om inloggningsskärmen inte dyker upp inom en minut så åtgärdar vi problemet.

Uppgift 1 – Binära tal

I filen `given_files/bits.cc` finns några testfall för denna uppgift som ska gå igenom när du är klar.

I denna uppgift ska du skapa en komponent som ska kunna användas för att hantera positiva binära tal. Skapa klassen `Bits` med följande medlemmar:

- Konstruktor. Tar emot en sträng innehållandes siffrorna 0 eller 1 som motsvarar det binära talet som ska lagras, exempelvis ”00100101”.
- `operator<<` för att skriva ut värden av typen `Bits` på en utström.
- `operator++`, både i prefix- och i postfixvariant för att stega det lagrade värdet med 1. Exempelvis ska 10011 stegas till 10100 och 10 stegas till 11.

Uppgift 2 – Hantera tal

Det finns en given indatafil, `data1.txt`, i mappen `given_files`.

I denna uppgift är det speciellt viktigt att använd standardbibliotekskomponenter i så hög grad som möjligt. Komplettera med lambdauttryck eller egna funktionsobjekt, ej vanliga funktioner, om det behövs för att lösa en uppgift. Inläsning och utskrift ska göras enligt följande.

- Programmet ska läsa indata från standardströmmen cin. Läsning från fil kan då göras genom omdirigering på kommandoraden, enligt följande exempel:

```
a.out < given_files/data1.txt
```

- Utdata ska skrivas på standardströmmen cout.

Programmet ska göra följande, i den ordning som anges nedan.

1. Läs indata i form av heltal (int) och lagra talen i en lämplig sekvenscontainer, med tanke på vad som ska göras nedan. Ingen ytterligare container får användas i programmet för att lagra och bearbeta data.
2. Skriv ut talen i containern, med ett mellanrumstecken efter varje värde.

```
149 tal har lästs in:
91 67 108 9 ... 68 82 68 13 31 74 4
```

3. Sortera talen i containern i stigande ordning.
4. Ta bort alla dubbletter, så att endast ett värde av varje finns kvar i containern.
5. Skriv ut de nu sorterade, unika talen i containern med ett mellanrumstecken efter varje värde.

```
Unika, sorterade tal:
1 2 3 4 5 6 7 ... 105 106 107 108 109 110 111
```

6. Beräkna hur många tal som motsvarar 5 procent. Vid beräkningen ska resultatet avrundas nedåt till närmast mindre heltal. Om det är 111 tal motsvarar 5% exakt 5,55 tal, vilket alltså ska avrundas till 5.
7. Ta bort de 5% minsta och de 5% största talen ur containern. Om containern till exempel innehåller 111 tal ska alltså de fem minsta och de fem största, sammanlagt tio tal, tas bort. Skriv sedan ut de återstående talen i containern, med ett mellanrumstecken efter varje värde.

```
Efter att de 5% minsta och 5% största talen tagits bort:
6 7 8 9 10 ... 102 103 104 105 106
```

8. Beräkna summan och medelvärdet av talen i containern och skriv ut dessa.

```
Summan av talen är 5656
Medelvärdet av talen är 56.0
```

Uppgift 3 – Böcker

I denna uppgift skall du designa en klasshirarki för skriven text. Det finns en given fil, `books.cc` i `given_files` med ett skelett och kommentarer som beskriver hur saker ska se ut.
Följande klasser ska du skall skapa:

Written_Text Är basklass. En skriven text har en författare, en titel samt funktioner för att komma åt dessa (utan att kunna ändra dem). Det skall inte gå att skapa en skriven text direkt, endast subklasser kan göra det.

Book Representerar en vanlig bok. Förutom författare och titel har en bok en bindning. Det skall bara gå att skapa en bok om alla tre anges. En bok har två undertyper, `Hardback` som alltid har sydd ("stitched") bindning och `Paperback` som alltid har limmad ("glued") bindning. Det skall gå att hämta ut bindningstypen från en bok.

Textfile representerar en text i elektronisk form. Den har utöver författare och titel ett filformat. Alla tre attribut måste anges för att skapa en klass av denna typ. Det skall gå att fråga en textfil efter dess format.

Magazine representerar slutligen en slags tidning. Författare och titel måste anges för att få skapa en instans.

Det ska finnas en funktion `str()` som ger en strängrepresentation av textens information. Denna ska vara polymorf.

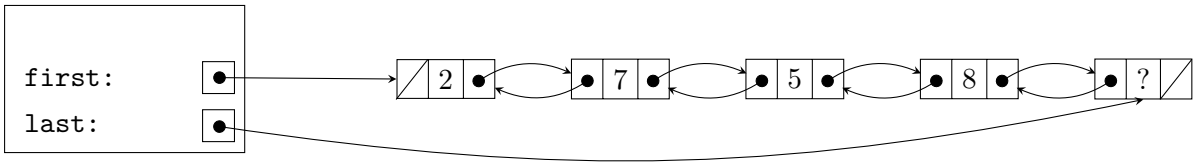
Uppgift 4 – Sortera lista

I katalogen `given_files` finns filerna `List.{cc,h}` som innehåller en implementation av en dubbellänkad lista (dvs varje nod har koll på både nästa och föregående). I slutet av listan finns en s.k. sentinelnod vars värde inte spelar någon roll (se figur nedan). Det finns en medlemsfunktion `sort` som (ganska ineffektivt) sorterar listan. Tyvärr använder den sig av en funktion `swap_nodes` som ännu inte finns. Det är din uppgift att implementera denna. Just nu anropas funktionen med två nodpekare, men behöver du fler värden från `sort` får du även modifiera funktionen. Observera att `swap_nodes` ska ändra på pekarnas värden, du får INTE byta plats på innehållet i noderna!

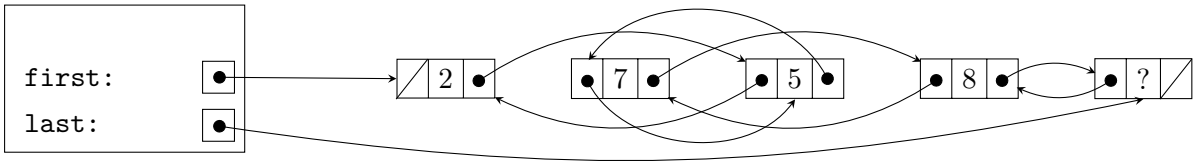
TIPS: Tänk på att sentinelnoden alltid ligger sist i listan (bör inte flyttas) och att första elementet kan behöva specialbehandlas.

Nedan följer två bilder som symboliserar vad som ska hända när två noder byter plats.

Du har även tillgång till några testfall i filen `given_files/list-test.cc`.



Figur 1: Lista innehållandes värdena 2,7,5 och 8



Figur 2: Lista innehållandes värdena 2,5,7 och 8

Uppgift 5 – Singla slant

En professor i statistik har en stor myntsamling. Varje mynt har ett årtal, ett nominellt värde (värdet skrivet på myntet) och ett försäljningsvärde (om det säljs idag). Dessutom kan ett mynt singlas (singla slant, coin toss) och ger då som resultat "krona" eller "klave" ("head" or "tail"). Professorn har upptäckt att 52% av befolkningen felaktigt tror att sidan med kungens huvud är "klave" och sidan med en krona "krona". Detta är fel eftersom "krona" syftar på begreppet "rikets styre", dvs kungen. Med detta som grund utarbetar nu professorn en strategi för slantsingling.
Professorn satsar alltid myntets nominella värde på att "krona" kommer upp. Om kungens huvud kommer upp övertygar han alltid motspelaren att denna förlorat genom att hänvisa till bevis (wikipedia mm). Om andra sidan kommer upp låter han motspelaren förlora om denna tror sig förlorat.

Skriv ett program som simulerar hur mycket professorn vinner på 1000000 slantsinglingar (med olika personer). Myntets information ska representeras som en klass.
Huvudprogrammet frågar användaren om de data som behövs för att skapa ett mynt och skickar myntet till en fristående funktion som gör beräkningen.

Körexempel 1

```
Mata in myntets nominella värde: 2
Mata in myntets försäljningsvärde: 5.67

Professorn vinner 972868.00 kr
```

Körexempel 2

```
Mata in myntets nominella värde: 0.25
Mata in myntets försäljningsvärde: 1.12

Professorn vinner 120631.50 kr
```