TDP004 - Tentamen

2019-04-26

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 studentklient.
- 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 ($+B$)

All bonustid gäller endast under den första ordinarie tentamen i samband med kursen (januari). Varje moment i kursen som ger bonus ger 5 minuter extra tid för högre betyg på sluttentamen, upp till maximalt 45 minuter. Detta är markerat med $+B$ i tabellen där bonus räknas.

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. Det betyder att för högre betyg måste du på sluttentamen lösa minst 2 uppgifter.

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 svenska flaggan 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 med grön skrivbordsbakgrund. Efter en stund kommer även din studentklient 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 din studentklient 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 *partiell* tillgång till referenssidorna på `http://www.cppreference.com/` via webbläsaren Chrome. Speciellt går det inte att komma åt language-delenav cppreference (d.v.s. om det står `language` i url:en). 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 studentklienten.

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 studentklient 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. Anmäl till assistent eller vakt om inloggningsskärmen inte dyker upp inom en minut så åtgärdar vi problemet.

Uppgift 1 - Felaktiga synonymer

Du har fått en beställning på ett program som skall ta emot ett ord och ge tillbaka ett synonym. Problemet är att kunden helt har misstolkat vad ett synonym är, denne tror att ett synonym är ett ord som stavas likadant med en bokstav i skillnad. Efter lite diskussion med kunden inser du att kunden verkligen vill ha det den säger.

I denna uppgift ska du använda standardalgoritmer för att skriva detta program. Det finns ett kodskelett för programmet i filen `given_files/synonym.cc`. Din uppgift är att implementera funktionerna `get_synonym` och `differs_at_one_letter`. Observera att inga loopar får användas och inte heller `std::for_each` utan du skall använda lämpliga standardalgoritmer. Det finns instruktioner för hur funktionerna kan tänkas att fungera i den givna filen.

Körexempel (användarinmatning i fet stil)

```
Mata in ett ord: barn
bark

Mata in ett ord: clownskor
clownskor

Mata in ett ord: panten
kanten
```

Uppgift 2 Stapla klot

Inledning

Vi tänker oss N stycken genomskinliga glasrör ståendes på rad. Som ett spel kan man tänka sig att vi släpper ner klot (orbs) från ovan och låter dem landa i rören. Vi tänker oss att det är slumpmässigt vilket rör som just varje klot hamnar i. Vi antar även att rören är väldigt långa och smala och att det kan finnas maximalt 99 stycken rör.

Funktionella krav

Skriv ett program där användaren får mata in N, d.v.s. hur många rör vi har. Användaren skall därefter mata in hur många klot vi vill släppa ned (också maximalt 99 stycken). Programmet skall därefter slumpa hur kloten hamnar i rören och rita ut detta på skärmen. I den utritade "bilden" skall även rören vara numrerade från 1 upp till N. Ingen felhantering av indata krävs.

Ickefunktionella krav

- Du ska generera slumptal enligt C++11-standard.
- Din lösning ska ha minst en lämplig funktion utöver `main`.
- Utdata ska matcha körexemplen till formatet (inte exakt eftersom data är slumpat).

Körexempel (användarinmatning i fet stil)

```
Enter N: 5
Enter orb count: 13
          o
     o    o
     o    o  o
     o    o  o
  o  o    o  o
  1  2  3  4  5

Enter N: 4
Enter orb count: 7
        o
        o  o
  o  o  o  o
  1  2  3  4

Enter N: 13
Enter orb count: 7
                        o
  o  o             o    o     o     o
  1  2  3  4  5  6  7  8  9 10 11 12 13
```

Uppgift 3 - Bildstorlek?

Som ni vet finns det olika komprimeringsmetoder för bilder. I denna uppgift ska ni skapa en klassheirarki för att representera bildfiler och sedan skapa ett program som läser in information om flera filer och skriver ut deras sammanlagda totalstorlek. Observera att beräkningarna nedan är helt tagna ur luften och troligen inte är i närheten av verklig storlek.

Funktionella krav

Ditt program ska be användaren mata in en eller flera bilder. En bild matas in på en rad på formatet `komprimeringsformat bredd höjd`. Användaren avslutar genom att mata in ett `q` på egen rad. Då ska samtliga bilders storlek skrivas ut.

Följande format ska godkännas:

1. BMP med storleksberäkning: `bredd * höjd`
2. JPG med storleksberäkning: `bredd * höjd * 0.2 + bredd + höjd`
3. PNG med storleksberäkning: `bredd * höjd * 1.5 - bredd - höjd`

Inga övriga bildformat stöds och om något annat format matas in ska ett felmeddelande skrivas ut.

Ickefunktionella krav

1. Problemet ska lösas med en klassheirarki med klassen `Image` som abstrakt basklass.
2. Varje bildformat representeras med en konkret klass.
3. Beräkningen för storleken av en viss bild ska göras med ett polymorfiskt anrop.

Körexempel

```
Enter one line for each image on the format "type width height". Exit with "q".

JPG 5000 3000
BMP 100 100
TIFF 300 300
!!! TIFF is an unsupported file format !!!
PNG 1000 1000
q

Total size: 4516000 bytes!
```

Tips

Du får anta att inmatning alltid är på korrekt format och att filformatet alltid är angivet med stora bokstäver.

Uppgift 4 - Formaterad utskrift

Kommunikation med användaren är bland det svåraste man kan göra som programmerare. I denna uppgift ska du skapa en klass som kan användas för att hjälpa programmerare att formatera text. I filen `given_files/format.cc` finns exempel på hur denna klass ska användas och fungera.

Funktionella krav

Skapa klassen `Formatter` med medlemsfunktioner enligt nedan där varje funktion returnerar en sträng med en viss förinställd bredd. Om strängen som skickas till funktionen är kortare än den inställda bredden ska den genererade strängen fyllas ut med mellanslag, annars returneras den inskickade strängen.

- Konstruktor som tar emot och sparar en bredd på den formaterade texten i variabeln `width`.
- `string adjust_left(string str)` returnerar en sträng med längd av minst `width` tecken där `str` är vänsterjusterad.
- `string center(string str)` returnerar en sträng med `str` centrerad.
- `string adjust_right(string str)` högerjusterar `str`.

Uppgift 5 - Produktsummering

Inledning

I `given_files/order_book.txt` finns en samling produktbeställningar. En rad i filen innehåller först ett produktnummer (1-7 siffror långt) och sedan ett antal. Alla raderna i filen har korrekt format.

Funktionella krav

Ett problem är att en sådan här fil kan innehålla många beställningar, så att samma produktnummer kan finnas på flera rader. Din uppgift är att skapa ett program som går igenom angiven fil en gång och skriver ut totalsumman beställningar för varje unikt produktnummer. Du måste alltså summera antalet för alla lika produktnummer, se exempel.

Utskriften ska vara i form av en snygg tabell sorterad enligt produktnummer, lägst först. Produktnumret ska vara vänsterställt och antalet högerställt.

Ickefunktionella krav

1. Namnet på filen med beställningar ska ges på kommandoraden, med alla tillämpliga felkontroller.
2. Lösningen ska använda en lämplig STL-container.
3. Filen får endast läsas igenom en gång.
4. Utdata ska matcha körexemplen till fullo.

Körexempel (användarinmatning i fet stil)

```
$ a.out
Usage: a.out FILE

$ ./a.out given_files/order_book.txt and-another-arg
Usage: ./a.out FILE

$ ./a.out given_files/no_such_file.txt
Error: 'given_files/no_such_file.txt' can not be opened!

$ ./a.out given_files/order_book.txt
PID    AMOUNT
13        123
16        164
22         16
31         25
```