TDP004 - Tentamen

2016-03-30

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 eller rökpaus 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. C++ Primer 5th ed.) En A4-sida 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, 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. 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++11` används för att kompilera med "alla" varningar *som fel*.
`w++11` används för att kompilera med "alla" varningar. **Rekommenderas.**
`g++11` används för att kompilera `utan` varningar.
`gccfilter e++11` används för att köra `e++11` genom `gccfilter`.
`gccfilter w++11` används för att köra `w++11` genom `gccfilter`.
`gccfilter g++11` används för att köra `g++11` genom `gccfilter`.
`valgrind --tool=memcheck` används för att leta minnesläckor.

C++ referenssidor

På tentan har du tillgång till referensmaterial från `http://cppreference.com/` och `http://www.cpluplus.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.

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 Delprodukter

Inledning

I en programmeringstävling ges följande problem.

Funktionella krav

Användaren matar in två heltal, N och T. Dessa matas in på en rad. På följande rad matas heltalen $n_1, n_2, ...n_N$ in (alltså exakt N heltal), och på tredje raden matas heltalen $t_1, t_2, ...t_T$ in (exakt T heltal). Nu ska programmet för varje t mata ut produkten av alla tal upp till n_t. Om $t = 5$ bildas alltså produkten av de 5 första talen n och om $t = 0$ anses produkten vara 1. Du kan anta att alla produkter ryms i en vanlig `long long int` och att alla tal matas in korrekt.

Ickefunktionella krav

- Din lösning ska använda minst en STL-behållare och minst en STL-algoritm. `std::vector` och `std::accumulate` rekommenderas.
- Utdata ska matcha körexemplen till fullo.

Körexempel (användarinmatning i fet stil)

```
5 9
1 2 3 4 5
1 2 3 2 1 0 4 5 3
1
2
6
2
1
1
24
120
6

13 3
3 8 7 6 5 4 2 3 9 6 7 3 9
13 4 7
1234517760
1008
40320
```

Uppgift 2 Kvittoskrivare

Inledning

En liten enmansfirma behöver ett program för att enkelt skapa kvitton. Ett kvitto består av ett antal produktrader. Varje produktrad har en produktbeteckning (ett ord) följt av ett styckpris (flyttal) och ett antal (heltal). Om antalet utlämnas antar vi det ska vara 1.

Funktionella krav

Du ska skapa ett program som låter användaren mata in produktrader. Inläsningen av ett kvitto avslutas med strömslut (Ctrl-D på tom rad). Därpå ska programmet mata ut ett formaterat kvitto med produktrader summerade och formaterade enligt körexemplet. Raderna ska vara sorterade efter produktbeteckning (bokstavsordning).

Ickefunktionella krav

1. Du ska använda en väl inkapslad klass för att representera ett kvitto. Det ska finnas en funktion för att lägga till en produktrad, och en funktion för att skriva ut det färdiga kvittot.

2. Nyckelordet `const` ska användas där det är lämpligt.

3. Om en redan inmatad produktbeteckning matas in en andra gång ska den senare produktraden ersätta den tidigare.

4. Priser matas in inklusive moms. Momsen är 20% av totalpriset.

5. Utdata ska matcha körexemplen till fullo. Produktbeteckningen skrivs ut med så många teckenpositioner att den längsta får plats. Priser skrivs ut med 9 teckenpositioner varav 2 decimaler. Antalet skrivs ut med 4 teckenpositioner.

Körexempel (användarinmatning i fet stil)

```
Vitrinskaop 1799 1
Schaeslong 795.0 2
Hurts 3999.0 1
Hyllplan 67.5 4
Hurts 399.0 2
Konsol 49.0 16
Vaeggfaeste 114.0 4

Hurts        399.00   2   798.00
Hyllplan      67.50   4   270.00
Konsol        49.00  16   784.00
Schaeslong   795.00   2  1590.00
Vaeggfaeste  114.00   4   456.00
Vitrinskaop 1799.00   1  1799.00
================================
Totalt ..............  5697.00
Varav moms ..........  1139.40
```

Uppgift 3 Wordfeud

Inledning

Wordfeud är ett mobilspel som går ut på att pussla ihop ord av de 7 bokstäver man har på handen. De två spelarna har en gemensam pool att dra nya bokstäver från när de lagt ett ord. De bokstäver som finns i poolen från start är lagrade i filen `given_files/WORDFEUD.TXT`. I filen finns först en bokstav och därpå antalet förekomster av den bokstaven i poolen. Så fortsätter det för hela alfabetet.

Funktionella krav

Du ska skapa en prototyp till den programmodul till Wordfeud som slumpar fram nya tecken till en spelare. Prototypen består av ett program som ber användaren mata in ett antal tecken som ska slumpas. Programmet slumpar sedan fram så många bokstäver från poolen, skriver ut dem och tar bort dem från poolen. Om fler tecken än som finns i poolen begärs slumpas så många tecken det går. När poolen är tom avslutas programmet.

Ickefunktionella krav

1. Du ska använda slumptalmekanismer enligt C++11-standard (du får *inte* använda `srand` eller `rand`).
2. Namnet på filen med startpoolen ska ges på kommandoraden, med alla tillämpliga felkontroller.
3. Din lösning ska ha minst en lämplig klass eller funktion utöver `main`.
4. Lösningen ska använda en lämplig STL-container.
5. Filen får endast läsas igenom en gång.
6. Utdata ska matcha körexemplen till formatet (inte exakt eftersom data är slumpat).

Körexempel (användarinmatning i fet stil)

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

$ ./a.out given_files/WORDFEUD.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/WORDFEUD.TXT
7
OAZAYOI
1
N
18
ACWBMFQALNUEDIIAEN
99
BKRRESRTAEWOAERDAXPSTRTIEMNOIIOOLUDADUTSSCHVGVAEIHLTPYEETLIUTHSOIERGNDJNFEGE
```

Uppgift 4 Kopieringsstatistik

Inledning

I C++ går det att skapa många objekt av varje klass. Dock är det svårt att manuellt hålla reda på när ett objekt kopieras och hur många olika kopior som kan tänkas finnas. Dock är språket så kraftfullt att det går att skapa en klass som automatiskt håller reda på detta via kopieringskonstruktor och tilldelningsoperator.

Denna klass skulle sedan enkelt kunna ärvas för att få statistik på hur ofta subklassen kopieras. I denna uppgift skippar vi dock den delen.

Funktionella krav

Skapa en klass som håller reda på hur många gånger den kopierats och hur många kopior som finns kvar. Detta åstadkoms lämpligen genom att klassen håller reda på två pekare. En som pekar ut antal kopieringar totalt, och en som pekar ut antal kvarvarande kopior. Vid grund kopiering kommer då kopian att peka på samma pekare som originalet och därmed kunna uppdatera räknarna för bägge. När den sista kopian försvinner ska totala antalet kopior som gjorts skrivas ut.

Ickefunktionella krav

1. Det givna huvudprogrammet ska fungera utan modifikation.
2. Inga minnesläckor ska förekomma.
3. Utskrifterna ska matcha körexemplen till fullo.

Körexempel

```
./a.out
0
-
10
-
0
1
-
2
3
```

Uppgift 5 Yatzypoäng

Inledning

Yatzy är ett tärningsspel där man slår med 5 tärningar. Utifrån resultatet väljer man sedan en av ett antal poängkategorier. Vald kategori kan inte väljas igen, så det gäller att välja en kategori som ger bra med poäng och som man inte tror man kommer behöva senare. Bland kategorierna finns bland andra `Triss` (poäng för de tre tärningar som visar lika), `Femmor` (poäng för alla femmor) och `Sexor` (poäng för alla sexor).

Funktionella krav

Skriv ett program som läser in ett Yatzy-slag (fem heltal) och skriver ut hur mång poäng det skulle ge enligt de tre påtalade kategorierna.

Ickefunktionella krav

1. Koden ska vara objektorienterad med en basklass `Score_Type`, två subklasser `Three_Of_A_Kind` och `Digit`. Till `Digit` ska finnas subklasserna `Fives` och `Sixes`.
2. Alla medlemmar ska vara placerade på lämplig nivå i hierarkin.
3. Klasserna ska ha en medlemsfunktion `score` som tar in ett slag (5 heltal) och returnerar hur många poäng slaget ger enligt klassens egen kategori. Det ska även finnas en medlemsfunktion `name` som skriver ut namnet på klassen (som kan hårdkodas i funktionen).
4. Huvudprogrammet ska skapa en vektor med de tre poängkategorierna `Fives`, `Sixes` och `Three_Of_A_Kind`. Vektorn ska sedan stegas igenom och med hjälp av polymorfi skriva ut respektive kategoris poäng för ett inmatat slag.
5. Utskrifterna ska matcha körexemplen till fullo.

Körexempel (användarinmatning i fet stil)

```
Enter 5 integers: 1 2 3 4 5
5p Fives
0p Sixes
0p Three of a kind

Enter 5 integers: 1 6 1 1 6
0p Fives
12p Sixes
3p Three of a kind

Enter 5 integers: 5 5 5 6 5
20p Fives
6p Sixes
15p Three of a kind
```