TDP004 - Tentamen

2017-01-11

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 ($+B$)

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++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 – Dela filer

Denna uppgift tillgodoräknas om du klarade en uppgift under förtentamen. Betyg kommer läggas in i tentasystemet en stund efter tentans start. Observera att detta endast gäller för betyg 3. Vill du ha ett högre betyg kan denna uppgift lösas som tredje uppgift.

Filer kan vara långa. Ibland önskar man att man kunde ta en bit i taget. I denna uppgift skall du skapa ett program som läser från en fil och skriver ut den i terminalen, i stycken.

Då man startar programmet skall man kunna ange två argument på kommandoraden. Det första argumentet är själva filnamnet på den fil som skall behandlas. Det andra argumentet (ett positivt heltal) anger hur många rader som varje stycke skall ha. Programmet skall sedan skriva ut varje stycke med en tydlig rubrik som innehåller styckesnummret (se körexemplet). Raderna i filen kan vara godtyckligt långa. Om antalet rader i filen inte är jämt delbart med antalet rader per stycke, så blir sista stycket helt enkelt kortare än de övriga. Filerna som vi testar med innehåller alltid minst en rad.

Körexempel 1:

```
>> ./a.out LITENFIL.TXT 5
 ========== STYCKE 1 ==========
Detta är en mycket
kort fil som passar
bra som exempel.
Den innehåller exakt
tretton rader, men
 ========== STYCKE 2 ==========
det betyder inte att
det är en otursfil.

Det finns dock även
en blank rad i den
 ========== STYCKE 3 ==========
här filen, så det
gäller att man inte
tänker fel på den.
```

Körexempel 2:

```
>> ./a.out LITENFIL.TXT 7
 ========== STYCKE 1 ==========
Detta är en mycket
kort fil som passar
bra som exempel.
Den innehåller exakt
tretton rader, men
det betyder inte att
det är en otursfil.
 ========== STYCKE 2 ==========

Det finns dock även
en blank rad i den
här filen, så det
gäller att man inte
tänker fel på den.
```

Filen `LITENFIL.TXT` finns i mappen `given_files`. Där finns även en längre textfil att testköra med.

Uppgift 2 – Resultatlista

Antag att vi har ett adminstrationsverkyg som håller koll på resultat från en given tenta. Detta system kan generera resultatlistor på följande format:

```
<NAMN>:<EFTERNAMN>:<PERSONNUMMER>:<RESULTAT>:<BETYG>
```

På filen `given_files/STUDENT_RESULTS.TXT` finns en fil med genererat (slumpat) utdata från detta tänkta system. Din uppgift är att kopiera filen `given_files/handle_results.cc` till din arbetskatalog och lägga till det som behövs för att skapa ett program som kan läsa in ett antal studentresultat från standard inmatningsström och skriva ut dem sorterade efter i första hand betyg (med ordningen 5,4,3,U), därefter efternamn (bokstavsordning) och till sist förnamn. Nedan visas ordningen för den givna filen (där `...` symboliserar ett flertal bortklippta rader). Utskriftsformatet ska vara enligt nedan.

```
Namn                        Personnummer  Betyg (Poäng)
Ahl, Jonatan                920828-XXXX       5 (20)
Almkvist, Therese           870225-XXXX       5 (24)
...
Ruggiu, Jonathan            891030-XXXX       4 (15)
Vilkancis, Adam             961102-XXXX       4 (17)
Akhtar, Antymos             960815-XXXX       3 (14)
Balzereit, Rikard           890617-XXXX       3 (13)
...
Wasing, Lena                950724-XXXX       U ( 2)
Zornic, Joanna              880731-XXXX       U ( 2)
```

OBS: Du får inte göra några ändringar i funktionen `main`.

Tips: För att slippa mata in alla studenter för hand kan du använda omdirigering av inmatningsströmen: `a.out < given_files/STUDENT_RESULTS.TXT`.

Uppgift 3 – Tidsintervall

Ett tidsintervall anges normalt sett med en starttid och en slutid. I denna uppgift har du en fil där varje rad i filen innehåller en starttid och annan fil där motsvarande rad är en sluttid för ett givet intervall. Din uppgift är att skapa ett program som hittar det längsta tidsintervallet och skriver ut detta på formatet HHMM.

Funktionella krav

- Namnen på filerna ska tas på kommandoraden. Om något går fel (exempelvis att något saknas eller att någon av filerna inte kan öppnas) ska ett bra felmeddelande skrivas ut och programmet avbrytas.
- Ett klockslag anges som ett heltal på formatet HHMM
- Om sluttiden är tidigare än starttiden ska det antas att sluttiden är nästa dygn. Exempelvis ska starttid 2359 och sluttid 0001 ge längden 0002

Filerna `START_TIME` och `END_TIME` i `given_files` visar några exempel på start- respektive sluttider.

Uppgift 4 – Spel

I denna uppgift ska du skapa en klassheirarki för att representera olika typer av spel. Här skulle man kunna hitta på flera olika typer, men för att begränsa uppgiften något begränsar vi oss till brädspel (alla spel som spelas på något typ av spelbräde) och kortspel. Följande klasser ska skapas:

Klassen `Game` ska vara en abstrakt klass och ska lagra spelets namn och antal spelare som kan delta i spelet. Antal spelare anges som två heltal (minsta och största antal spelare). Dessutom ska klassen ha följande medlemsfunktioner:

name returnerar namnet

print ska skriva ut en beskrivning av spelet på en given ström.

`Game` har två direkta subklasser, `Board_Game` och `Card_Game`. `Board_Game` måste förutom de data Game kräver även hålla koll på spelbrädets storlek (två heltal).

Vissa brädspel är kooperativa, dvs att spelarna samarbetar emot spelet istället för mot varandra. Vissa kooperativa spel har förrädare - en spelare som utan att avslöja sig försöker förstöra för de övriga spelarna. Detta ska representeras med en klass `Cooperative` som en subklass till `Board_Game`. Denna klass måste hålla koll på om det finns möjlighet till en förrädare. Normalt sett ska det inte finnas förrädare.

Kortspel kan antingen använda en vanlig kortlek eller ha speciella kort. Normalt sett ska `Card_Game` anta att spelet är med en vanlig kortlek, men detta ska gå att ange.

Funktionella krav

- Utskriften av programmet ska bli enligt kommentarer i den givna filen.

Ickefunktionella krav

- Klasserna beskrivna i inledningen ska implementeras korrekt.
- `const` ska användas korrekt.
- Programmet får inte ha minnesläckor eller onödig upprepning.

Uppgift 5 – Pi

Du ska i denna uppgift skapa ett program som beräknar en uppskattning av PI enligt följande algoritm:

1. Skapa två vectorer (härmed kallade X och Y) med förallokerad plats för N double-värden.
2. Fyll de båda vectorerna med uniformt spridda slumpvärden i intervallet $[-1.0, 1.0]$.
3. Skapa en ny tom vector, härmed kallad `C`, för att lagra bool-värden.
4. Använd algoritmen `std::transform` för att, för varje koordinat (x,y) som skapas genom att parvis ta element från vectorerna X och Y, lägga in värdet `true` i C om koordinaten ligger i enhetscirkeln (en cirkel med radie 1 som ligger i origo), dvs att $x^2 + y^2 < 1$. Om koordinaten inte ligger i enhetscirkeln ska `false` sparas i C.
5. Beräkna antalet koordinater (F) som fanns inom enhetscirkeln genom att använda `std::accumulate` med vector C.
6. Uttrycket $4 * (F/N)$ ger nu en uppskattning av Pi. Skriv ut detta värde.

Prova din algoritm med olika värden på `N`. Med $N = 1000000$ bör du få ungefär två decimalers noggrannhet. Det ska vara enkelt att ändra värdet på N i ditt färdiga program.