TDP004 - Tentamen

2017-04-19

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 – Komprimering

Alla har vi någon gång använt verktyg för att komprimera data (exempelvis zip eller gzip). I denna uppgift ska du implementera en komprimeringsalgoritm med hjälp av standardbiblioteket.

Algorimen bygger på att byta ut vanligt förekommande ord i en text mot en kortare symbol och därmed minska antal tecken i texten.

1. Läs in samtliga ord från STDIN till en `vector<string>` (härmed kallad `words`) med hjälp av `istream_iterator`.
2. Skapa en frekvenstabell över orden i `words` med hjälp av en `std::map<string, int>`.
3. Kopiera elementen i frekvenstabellen till en `vector<pair<string, int>>`, `v`.
4. Sortera `v` efter frekvensen i sjunkande ordning med hjälp av `std::sort` och ett eget lambda-uttryck.
5. Kopiera de 10 oftast förekommande orden till en `vector<string>`, `replace`.
6. Byt ut de ord i `words` som finns i `replace`. Ett ord som finns i `replace` byts ut mot symbolen `$N` om `N` är ordets index i `replace`. De tio orden byts alltså ut mot symbolerna `$0` till `$9`. Detta kan exempelvis lösas med `std::transform` med ett eget lambda, men det är ok att använda en loop här.
7. Skriv ut en översättningstabell på formatet `$0=ord1;$1=ord2;...;$9=ord10;` på en rad på cout, exempelvis med hjälp av `std::transform` och en `ostream_iterator`.
8. Skriv ut orden i `words` separareade med mellanslag. Observera att vi här kommer förlora andra vita tecken som eventuellt fanns i ursprungstexten.

I katalogen `given_files` finns ett program `uncompress` som gör det motsatta arbetet ovanstående algoritm utför. Du kan alltså använda det programmet för att se om din lösning stämmer.

```
cat textfil.txt | a.out | given_files/uncompress
```

Bör ge innehållet i filen `textfil.txt` som utskrift (utan radbrytningar) om du gjort rätt.

Du kan anta att symbolerna `$0..$9` inte förekommer i ursprungstexten samt att texten har minst 10 unika ord.

Uppgift 2 – Sökväg

Absoluta sökvägar i UNIX-liknande system utgår ifrån systemets rot (`/`). Roten innehåller flera kataloger vilka i sin tur kan innehålla nya kataloger och filer. En katalog har alltid en förälder (roten anses här ha sig själv som förälder). En fil kan ha en filändelse (tecknen efter sista punkten i filnamnet). Din uppgift är att skapa en klass med namn `Path` för att representera en sökväg.

Funktionella krav

Följande medlemsfunktioner och operationer ska stödjas av din klass

Konstruktor Det ska gå att skapa sökvägar med en sträng. Om strängen avslutas med ett snedstreck (`/`) anses sökvägen vara en katalog, annars en fil. Om ingen sökväg anges ska objektet representera roten.

is_dir Ger `true` om sökvägen är en katalog, annars `false`.

is_file Ger `true` om sökvägen är en fil.

parent Ger en *sökväg* till förälderkatalogen (katalogen filer ligger i om sökvägen är en fil).

basename Sista delen av sökvägen som en sträng. Dvs filnamnet om det är en fil eller namnet på nedersta katalogen.

contains(other) Ger `true` om sökvägen `other` ingår i nuvarande sökväg. Dvs om man kan stega nedåt i filsystemet för att nå `other`. Om nuvarande sökväg är en fil ska undantaget `std::runtime_error` kastas.

extension Ger filändelsen som en sträng. Detta ska endast fungera för filer och kan vara tom. Om den anropas för en katalog ska ett `std::runtime_error` kastas.

Typomvandling en sökväg ska kunna konverteras till en `std::string` automatiskt.

Filen `given_files/path_test.cc` innehåller catch-baserade testfall för din klass som bör fungera.

Uppgift 3 – Tentaresultat

På filerna `given_files/student.{h,cc}` finns en representation av resultatet på en tenta för en student tillsammas med några funktioner för att hantera detta resultat. Denna är dock implementerad på ett C-liknande sätt och din uppgift är därför att göra om detta till en klass med god inkapsling. När du är klar ska programmet i `given_files/test_student.cc` fungera utan ändringar.

Uppgift 4 – 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 samanlagda 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`
3. PNG med storleksberäkning: `bredd * höjd * 1.5`

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: 4510000 bytes!
```

Uppgift 5 – Extra O

Du ska skapa ett program som låter användaren mata in ett antal ord. Din uppgift är att för varje ord som användaren matar in skriva ut ordet något förändrat. Du ska modifiera ordet enligt följande algoritm:

1. Räkna ut hur många av ordets tecken som inte är bokstaven o. Detta antal kallas nedan `N`.
2. Hitta sista o:et i ordet.
3. Stoppa in `N` nya o:n efter sista o:et.

Körexempel:

```
Mata in ord: Några ord modifieras
Modifierade: Några ooord mooooooooooodifieras
Mata in ord: till godo
Modifierade: till godooo
Mata in ord: <CTRL-D>
```

Ickefunktionella krav

1. Du ska här visa god användning av medlemsfunktioner av `std::string` och standardbiblioteket.
2. Du får ha en loop för huvudprogrammet (läs in rad + skriv ut). Utöver det får du inte ha några egna loopar eller andra typer av iterationer (exempelvis rekursion).