TDP007 - Dugga 1

2023-02-01

Regler

- All kod efterrättas på denna dugga
- 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 start.
- Vid toalettbesök ska vakt informeras.
- 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.

Hjälpmedel	En Rubybok (exempelvis the pickaxe Ett A4-ark med egna anteckningar Tillgång till webresurser: ruby-docs, rubular och tidig version av kursboken

Information

Betygssättning vid tentamen

Tentamen består av ett antal uppgifter på varierande nivå. Uppgifter som uppfyller specifikationen samt följer god sed och konventioner krävs för maxpoäng på en uppgift. Avvikelser från ovanstående ger avdrag.

Duggan består av fyra uppgifter (några indelade i deluppgifter) som totalt kan ge 25 poäng. Dessa poäng summeras ihop med poängen från den andra duggan.

Inloggning

Logga in på datorn i labbsalen med ditt liu-id och lösenord. Detta är samma inloggningsuppgifter som du använder i Lisam.

Skrivbordsmiljön

När du kommit in i tentasystemet har du en normal skrivbordsmiljö (Mate-session i Ubuntu). 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

`ruby` används för att köra en ruby-fil.
`irb` används för att starta ruby i interaktivt läge.
`ri` används för att komma åt den inbyggda dokumentationen.

Ruby-docs

På tentan har du tillgång till referenssidorna på `https://ruby-doc.org/` (både core och stdlib finns tillgänglig) 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. Tag hjälp av assistent i sal om det inte fungerar.

Rubular

På tentan har du tillgång till sidan `https://rubular.com/`.

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.

Avslutning

När du skickat in alla uppgifter och känner dig färdig kan du logga ut och lämna salen. Antalet poäng meddelas i efterhand via webreg.

Avsluta alla öppna program och logga ut ur datorn. Lämna inte datorn innan du ser att du är utloggad.

Uppgift 1 - Teori (6p)

- Ge ett bra exempel (som inte finns i boken) på hur man använder metoden `Array#inject` och förklara vad som händer när man kör det. (1p)
- När ska man använda strömparsning (SAX) och när ska man använda trädparsning (DOM)? Ge ett exempel på användningsområde och förklara kort varför den ena metoden skulle kunna vara bättre. (1p)
- I kursen har vi arbetat med regexp, strömparsning och trädparsning hitills för att lösa uppgifter. I samband med dessa uppgifter förväntades du producera testfall som försäkrade implementationens funktionalitet och kommunicerade denna till andra studenter. Denna typ av implementationer kan vara utmanande att testa eftersom de beror på filer som kan se ut på många olika sätt. Argumentera för hur man kan testa en strömparser(SAX). (4p)

Uppgift 2 - SAXparsning (7p)

I filen `news.xml` finns delar av nyhetsflödet för ett spel på platformen Steam. Ditt jobb är att med en SAX-parser läsa ut de nyheter som innehåller elementet `<tags>` till en array. Denna array skall innehålla hashar där titeln av nyheten framgår samt det innehåll som finns för nyheten. Det finns 4 nyheter som har elementen `tags` i filen, se körexemplet.

Krav: Det är viktigt att vi i denna uppgift tar hänsyn till att det inte finns en xsd eller dtd att utgå ifrån. Vi vet således inte om andra filer av denna typ kommer ha elementen i samma ordning som i denna fil, eller om det kan finnas andra element än de vi ser här. För full poäng så ska din lösning vara robust nog att endast läsa ut det som specifikationen efterfrågar.

Tips: Det vanliga skelettet för sax-parsers finns i `sax_example.rb`

```
{"title"=>"Team Fortress 2 Update Released", "content"=>"An update to Team
  Fortress 2 has been released. The update will be applied automatically
  when you restart Team Fortress 2. The major changes include: Fixed
  disguised/cloaked..."}

{"title"=>"Team Fortress 2 Update Released", "content"=>"An update to Team
  Fortress 2 has been released. The update will be applied automatically
  when you restart Team Fortress 2. The major changes include: Fixed not
  being able to..."}

{"title"=>"Team Fortress 2 Update Released", "content"=>"An update to Team
  Fortress 2 has been released. The update will be applied automatically
  when you restart Team Fortress 2. The major changes include: Fixed
  movement bug..."}

{"title"=>"Team Fortress 2 Update Released", "content"=>"An update to Team
  Fortress 2 has been released. The update will be applied automatically
  when you restart Team Fortress 2. The major changes include: Fixed Your
  Eternal..."}
```

Uppgift 3 - DOMparsning (7p)

I filen foobar.xml finns information om alla anställda på företaget Foobar Inc. Alla anställda måste tillhöra en grupp och en av personerna i gruppen är chef. Grupperna är organiserade i olika divisioner. Det finns fem divisioner och några av dem är så små att de bara innehåller en enda grupp. En division är så stor att den i sin tur är uppdelad i två divisioner.

Din uppgift är att skriva olika funktioner för att kunna läsa in och hantera informationen i filen. Dy ska lösa detta problem med DOM-parsning du hittar lite information om metoderna i klassen Element i filen `rexml.txt`.

Uppgiften består av flera deluppgifter som ger olika många poäng.

- (2p) Gör en funktion som beräknar och skriver ut medellönen i varje grupp. Exempel:

```
>> average_salary(doc)
Domestic: 33340
Overseas: 40250
Alpha: 35800
Beta: 35433
...
```

- (3p) Gör en funktion som tar in namnet på en person och returnerar namnet på den personen som är ledare för personens grupp. Exempel:

```
>> find_leader(doc,"Jessie Poarch")
=> "Benita Bisbee"
>> find_leader(doc,"Ted Bogar")
=> "Ted Bogar"
```

- (2p) Lönekontoret vill veta hur mycket pengar det måste betala ut varje månad. Skriv en funktion som summerar ihop månadslönerna för alla personer i företaget och dessutom lägger på 53% (för arbetsgivar- och pensionsavgifter). Exempel:

```
>> total_salary(doc)
=> 2866761.0
```

Uppgift 4 - Praktisk, Parsa text (5p)

- I filen tddi16.txt finns en kursplan för en kurs som ni kommer att läsa lite senare. På olika ställen i texten finns det kurskoder (se exempel på kurskoder i filen kurskoder.txt). Skriv en funktion som läser in filen och använder ett reguljärt uttryck för att matcha och hitta alla kurskoder. Svaret ska vara en array med unika kurskoder, sorterade i bokstavsordning. (2p)

```
>> find_codes("tddi16.txt")
=> ["TADI03", "TDDI04", "TDDI12", "TDDI16", "TDIU01", "TDP004", "TDP015"]
```

- I filen film.txt finns korta beskrivningar av filmer. Skriv en funktion som läser in filmbeskrivningarna och med hjälp av reguljära uttryck hittar alla namn som förekommer. Vi vill som svar ha någon form av datastruktur som innehåller vilka namn som förekommer i varje film. Det skulle t.ex. kunna se ut som nedan. (3p)

```
>> read_movies("film.txt")
=> [["Mannen med den gyllene pistolen",
       ["Francisco Scaramanga", "James Bond"]],
     ["The Fantastic Four", ["Doctor Doom", "Reed Richards",
         "Mr Fantastic", "Sue Storm", "Invisible Girl",
         "Johnny Storm", "Ben Grimm"]],
    ["Rosa Pantern slår igen", ["Charles Dreyfus", "Jacques Clouseau"]]]
```