TDP007 - Dugga 1

2020-02-13

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 (7p)

- (1p) Om man kör följande Ruby-program, varför skriver det ut att a är 2 och inte 4?

```
def sub
  $a = 2
  yield
end

def main
  $a = 4
  sub { puts "$a = #{a}" }
end

main
```

- (2p) Du har tidigare arbetat i minst 2 andra programmeringsspråk. Beskriv två betydelsefulla sätt som Ruby skiljer sig från ett valfritt programmeringsspråk och motivera vilka för- och/eller nackdelar som finns med dessa skillnader
- (2p) Skapa ett regex som matchar ett svenskt personnummer givet på formatet: YYYYMMDD-XXXX. För full poäng skall begränsad rimlighetskontroll av siffrorna innan bindestrecket utföras (m.h.a. regex). En dag kan inte vara större än 31 eller mindre än 1 etc.
- (2p) Förklara när `sax-parsning` är det föredragna sättet att parsa dokument på. Förklara kortfattat vad sax-parsning är och hur man använder det.

Uppgift 2 - Praktiskt (4p)

I den bifogade filen 2.rb finns klassen Boardgame och ett litet huvudprogram. Klassen sparar lite data om ett brädspel och vet hur den konverterar sig till en sträng. Men vi vill också kunna sortera denna data med den inbyggda funktionen `.sort`. För att det ska fungera behöver vi kunna jämnföra brädspelen med varandra, detta beteende specificeras i ruby m.h.a spaceship operator `<=>`.

- (2p) Titta på dokumentationen för den mixin som heter ”Comparable”(det är en modul) och lägg till funktionaliteten som krävs för att objekt av typen Boardgame ska kunna jämföras med varandra. Efter du lagt till denna funktionalitet ska brädspelen i arrayen kunna sorteras med medlemsfunktionen `.sort!` i stigande ordning baserat på instansvariabeln `weight`.
- (2p) Utforma därefter en uppsättning lämpliga enhetstester för klassen Boardgame.

Uppgift 3 - Praktisk, XML (8p)

BGG eller BoardGameGeek är ett mecka för bräd-spelsälskare av alla dess slag. BGG tillhandahåller (som många sidor) ett API för den som vill ha ut information om sidan. I denna uppgift tillhandahåller jag filen game699.xml som berör det klassiska spelet HeroQuest som hämtats från BGGs API. Du kan med fördel öppna filen i en webbläsare för att enklare läsa den.

- (5p) Din uppgift är att parsa denna fil med en XML-parser och representera följande information i en lämplig datastruktur. Om du vill göra en `SAX-lösning` hittar du ett exempel att utgå från i filen `sax_example.rb` och om du vill göra en DOM-lösning hittar du lite information om metoderna i klassen REXML::Element samt några exempel på XPath-uttryck i filen rexml.txt. Sedan ska du skriva ut följande information enligt formatet i körexemplet nedan.
  Du behöver ta reda på följande information:
  - Spelets namn
  - En lista över spelets expansioner (alla boardgameexpansion”)

  Körexempel:

```
$ ruby 3.rb
> Game: HeroQuest
> Expansions:
> 1: Adventure 1: The Mountain Keep (fan expansion to HeroQuest)
> 2: Adventure 2: Slaves Of Zargon (fan expansion for HeroQuest)
> 3: Adventure 3: The Lost Books (fan expansion for HeroQuest)
> 4: Adventure 4: In The King's Service (fan expansion for HeroQuest)
> 5: Adventure 5: Beyond Grin's Crag - Kellar's Keep 2 (fanexpansio...
> 6: Adventure 6: Resurrection - Return of the Witch Lord 2 (fan ex...
> 7: Adventure 7: The Rescue of Princess Millandriell (fan expansio...
> 8: Adventure 8: The Horror Inside the Ancient Halls of Sunca (fan...
> 9: Heroquest: A Growl of Thunder
> 10: HeroQuest: Against the Ogre Horde
> 11: HeroQuest: Barbarian Quest Pack
> 12: HeroQuest: Dungeons of Peril (fan expansion for HeroQuest)
> 13: HeroQuest: Elf Quest Pack
> 14: HeroQuest: Kellar's Keep
> 15: HeroQuest: Return of the Witch Lord
> 16: HeroQuest: Running the Gauntlet
> 17: HeroQuest: Wizards of Morcar
> 18: Masters' Series: Adventure 1 - The Deadly Hand of Zargon (fan ...
```

- (3p) Ta reda på vilket antal spelare som anses vara det bästa antalet för spelet. Beräkna detta utifrån `<poll name=sugested_numplayers ...>`. Det antalet spelare som har flest röster i `result` elementet som har attributet value satt till `Best` är det bästa antalet spelare. Om du också löst första delen av uppgiften ska körexemplet se ut så här:

```
$ ruby 3.rb
> Game: HeroQuest
> Best with 5 players
> Expansions:
> 1: Adventure 1: The Mountain Keep (fan expansion to HeroQuest)
> 2: Adventure 2: Slaves Of Zargon (fan expansion for HeroQuest)
...
```

Uppgift 4 - Praktisk, Parsa text (6p)

I spelet TI4 (Twilight Imperium 4) kämpar många olika riken om att styra galaxen. Upp till 6 spelare väljer ett av dessa riken och spelar sedan mot varandra. I filen `ti4_data.txt` finns resultat som är hämtade från ett forum där det diskuteras vilket rike som är det bästa i hopp om att skapa en så kallad tier list.

Din uppgift är att läsa in filen och med ett regex läsa ut informationen du behöver. Du ska sedan representera varje rikes namn och deras vinst-procent i en passande databehållare. Sedan ska du skriva ut rikena sorterat enligt deras vinst-procent formatterat enligt körexemplet nedan. Kolumnerna i filen innehåller följande information i ordning från vänster till höger: namn, antal spelade spel, antal vinster, antal lika, antal förluster.

Körexempel:

```
$ ruby 4.rb
> Universities of Jol-Nar : 0.28
> Yin Brotherhood         : 0.27
> Federation of Sol       : 0.27
> Naalu Collective        : 0.26
> Emirates of Hacan       : 0.25
> Clan of Saar            : 0.25
> Nekro Virus             : 0.21
> Embers of Muaat         : 0.21
> Arborec                 : 0.21
> Yssaril Tribes          : 0.21
> Ghosts of Creuss        : 0.2
> LIZIX Mindnet           : 0.2
> Winnu                   : 0.19
> Mentak Coalition        : 0.18
> Barony of Letnev        : 0.17
> Xxcha Kingdom           : 0.14
> Sardakk Norr            : 0.09
```

(i Körexemplet blir 2 av procentsatserna avrundade till en siffra efter punkten. Om du skriver ut den andra decimalsiffran är det också ok. Du behöver inte göra kolumnjusteringen på samma sätt som den är gjord ovan.)