TDP007 - Dugga 1

2024-01-30

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 (5p)

Teorifrågor lämnas in som en uppgift i form av en textfil (.txt, .md, .org), det ska vara läsbart om man skriver ut filen i terminalen. Samla gärna alla frågorna i en och samma fil.

1.1 (2p)

Vad är viktigt att tänka på när man utformar enhetstester för att testerna ska vara effektiva?

1.2 (1p)

Vad är skillnaden på blackbox och whitebox testing samt någon för och nackdel med dessa?

1.3 (2p)

Kör nedanstående program, skriv ned resultatet och förklara varför det blev så. Koden finns i den givna filen u1.3.rb om du har problem med att kopiera koden.

```
def foo
  c = 2
end

def main
  a = 1
  begin
    b = 2
  end
  puts a if defined?(a)
  puts b if defined?(b)
  puts c if defined?(c)
end

main
```

Uppgift 2 - Regexp (6p)

I den givna filen `candidates_tournament_2022.txt` finns resultat från en shackturnering. Din uppgift är att läsa in filen och med ett regex läsa ut informationen du behöver. Du ska representera varje spelares namn, hur många poäng spelaren tog som vit, som svart samt totalt antal poäng i en associativ databehållare (`Hash`). Första namnet på en rad är spelaren som spelade som vit, det andra namnet är den som spelade som svart. 1 - 0 indikerar att vit vann, 0 - 1 att svart vann 0.5 - 0.5 att det blev remi(oavgjort).

Körexempel:

```
Name              W    B   Total
--------------------------------
Nepomniachtchi  4.5  5.0   9.5
Ding            3.5  4.5   8.0
Radjabov        4.0  3.5   7.5
Nakamura        5.5  2.0   7.5
Caruana         3.5  3.0   6.5
Firouzja        3.0  3.0   6.0
Rapport         2.5  3.0   5.5
Duda            4.0  1.5   5.5
```

Uppgift 3 - SAXparsning (7p)

Kunskapen vi som människor idag har inom diverse forskningsfält finns i stor utsträckning publicerad i form av artiklar. I den givna filen `infedu.xml` finns en samling av artiklar som utgjorde en tidning som berörde konceptet abstraktion inom datorvetenskapen samt en artikel som är förordet till hela tidningen.

Din uppgift är att med en SAXparser läsa ut titeln och författarna på alla artiklar(`<article>`) som listas under elementet `<articles>`. Det är totalt 8 artiklar du bör läsa ut information från. Varje artikel skall antingen representeras av en `Hash` eller en klass. Artiklarna skall sedan lagras i en `Array`. Sist men inte minst ska artiklarna skrivas ut enligt körexemplet, det finns en del speciella tecken i författarnas namn, du behöver inte oroa dig för eventuella teckenkodningsfel (och du kan eventuellt upptäcka något sådant fel i körexemplet):

Körexempel:

```
Teaching Software Engineering using Abstraction through Modeling
Mohsen DORODCHI, Nasrin DEHBOZORGI, Mohammadali FALLAHIAN, Seyedamin
POURIYEH

A Comparison of Abstraction and Algorithmic Tasks Used in Bebras Challenge
Jirí VANÍCEK, Václav ŠIMANDL, Patrik KLOFÁC

The Non-Deterministic Path to Concurrency - Exploring how Students
Understand the Abstractions of Concurrency
Filip STRÖMBÄCK, Linda MANNILA, Mariam KAMKAR

A Necessity-Driven Ride on the Abstraction Rollercoaster of CS1 Programming
Marco SBARAGLIA, Michael LODI, Simone MARTINI

Abstraction in Computer Science Education: An Overview
Claudio MIROLO, Cruz IZU, Violetta LONATI, Emanuele SCAPIN

Understanding Students Failure to use Functions as a Tool for Abstraction -
An Analysis of Questionnaire Responses and Lab Assignments in a CS1 Python
Course
Pontus HAGLUND, Filip STRÖMBÄCK, Linda MANNILA

Abstraction, Declarative Observations and Algorithmic Problem Solving
David GINAT

Tool-Aided Learning of Code Reasoning with Abstraction in the CS Curriculum
Megan FOWLER, Jason HALLSTROM, Joseph HOLLINGSWORTH, Eileen KRAEMER, Murali
SITARAMAN, Yu-Shan SUN, Jiadi WANG, Gloria WASHINGTON
```

Uppgift 4 - DOMparsning (7p)

Om man ägnar sig åt forskning inom något område så spenderar man mycket tid med att försöka hitta relevant forskning som andra gjort som man kan använda som bakgrund i sina egna artiklar. Det arbetet handlar mycket om att titta på nyckelord och läsa så kallade `abstracts` (korta sammanfattningar av artiklar).

I den givna filen `infedu.xml` finns artiklarna från en tidning (samma som i uppgift 3). Du ska skapa en funktion (och tillhörande huvudprogram) som heter `filter_keyword` denna funktion ska ta ett xml-dokument(`REXML:Document` eller motsvarande) och en sträng. Funktionen skall sedan filtrera artiklarna i xml-filen och returnera en `Array` av de artiklar som matchar kriterierna. Varje artikel skall representeras av en associativ behållare eller klass och innehålla artikelns rubrik och abstract.

Följande filtrering skall utföras:

- Alla artiklar som inte har attributet `article-type` satt till **article** ska filtreras bort
- Alla artiklar som inte har ett nyckelord som matchar strängen som skickades med till funktionen skall filtreras bort. Nyckelorden hittar du i varje artikel i elementet `<kwd-group>`, varje nyckelord finns i elementet `<kwd>`.

Körexempel (angivet nyckelord: computer science education):

```
Abstraction in Computer Science Education: An Overview
When we "think like a computer scientist", we are able to systematically...

Understanding Students Failure to use Functions as a Tool for...
Controlling complexity through the use of abstractions is a critical part...
```

I körexemplet har utskriften trunkerats för att göra exemplet tydligare, du kan i din lösning skriva ut hela rubriken och hela abstract.