TDP007 - Tentamen

2025-03-26

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 3 delar. Del 1 är uppgifter på grund-nivå som relaterar till tekniker från kursens första två seminarier, dessa bedöms som G/U. Del 2 är uppgifter på grund-nivå som relaterar till tekniker från kursens andra två seminarier, dessa bedöm som G/U. Del 3 består av mer avancerade uppgifter, där varje uppgift relaterar till någon teknik från kursens fyra seminarier. Uppgifter på del 3 ger upp till 10 poäng. Betyg på tentamen ges enligt följande tabell. För alla betyg så måste förutom de insamlade poängen alla uppgifter på Del 1 och Del 2 lösas med godkänt betyg.

Insamlade poäng Del 3	**Betyg**
10	3
17	4
24	5

Duggor

Den första duggan består av två delar. Del 1 är uppgifter på grundnivå som relaterar till tekniker från kursens första två seminarier och bedöms som G/U. Del 2 består av en mer avancerad XML-uppgift som ger upp till 10 poäng. Om alla uppgifter på Del 1 har godkänt betyg från denna dugga så har du Del 1 på ordinarie tentamen tillgodo.

Den andra duggan består av två delar. Del 1 är uppgifter på grundnivå som relaterar till tekniker från kursens sista två seminarier och bedöms som G/U. Del 2 består av en mer avancerad parser-uppgift som ger upp till 10 poäng. Om alla uppgifter på Del 1 har godkänt betyg från denna dugga så har du Del 2 på ordinarie tentamen tillgodo.

Om del 1 är avklarad på båda duggorna och du har samlat in totalt 14 poäng från duggorna så har du förutom Del 1 och Del 2 på ordinarie tentamen tillgodo också betyg 3 på tentamensmomentet.

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 std-lib 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.

Del 1: Uppgift 1 - SAX-parser

I filen `large_shipments.xml` finns data för paket som skickas från och till olika platser. Varje paket har också ett id. Ditt jobb är att med en Strömparser (SAX) hämta ut en lista över alla id på paket som skickas från Uppsala. Listan ska sedan returneras från parsern på något sätt och skrivas ut (se funktionella krav nedan). Exakt formattering av utskriften spelar ingen roll, men listan behöver innehålla korrekt data.

Funktionella krav:

```
["P147", "P178", "P125", "P238", "P114", "P168", "P137", "P200"]
```

Del 1: Uppgift 2 - DOM-parser

Lös samma problem som i Uppgift 1, men använd nu en DOM-parser istället för en SAX-parser. Använd minst ett xpath-uttryck.

Funktionella krav:

```
["P147", "P178", "P125", "P238", "P114", "P168", "P137", "P200"]
```

Del 2: Uppgift 3 - DSL

I filen `dsl.rb` finns data över vilka personer som är tillgängliga vilka dagar under 2 veckor. I den givna filen `uppgift3.rb` finns en påbörjad läsare för detta DSL. Du ska modifiera denna läsare så att den med `method_missing` läser ut information från filen som sedan skrivs ut enligt de funktionella kraven. Veckodagarna behöver hanteras med `method_missing`, veckorna kan hanteras med en separat funktion. Formatteringen av utskriften spelar ingen roll i denna uppgift, inte heller datatyperna, det behöver bara vara läsbart och innehålla rätt information. Siffrorna som hör till veckodagarna är antalet personer som anmält tillgänglighet vid respektive dag.

Funktionella krav:

```
{
  "34"=>{:monday=>4, :tuesday=>6, :wednesday=>5, :thursday=>6, :friday=>5},
  "35"=>{:monday=>3, :tuesday=>5, :wednesday=>5, :thursday=>4, :friday=>4}
}
```

Del 2: Uppgift 4 - LogikParser

I `uppgift4.rb` finns en påbörjad logikparser. I denna parser ska användaren kunna mata in följande 4 saker: `F` (false), `T` (true), `&` (and), `=` (==, likhet). Skriv om den givna parsern så att det givna huvudprogrammet ger rätt output:

Funktionella krav:

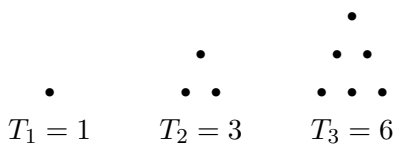
```
T = F >>> false
T = T >>> true
T = T & T >>> true
T = F = T & T = T = T >>> false
```

Tips: Ruby ser inte `false` och `true` som en gemensam typ utan som `FalseClass` respektive `TrueClass`. Kan vara viktigt vid utformningen av regeln som motsvarar `:atom`.

Del 3: Uppgift 5 - Icke-deterministisk programmering

I den givna filen `uppgift5.rb` finner du den AMB-klass som vi arbetat med under föreläsningsserien. Din uppgift är att använda denna för att lösa följande problem.

Ett `triangeltal` är ett tal som motsvarar antalet objekt som kan arrangeras i en liksidig triangel. Det n:te triangeltalet ges av summan av de n första positiva heltalen, vilket kan visualiseras som en växande triangel av punkter.



$T_1 = 1$ $T_2 = 3$ $T_3 = 6$

Matematiskt så kan detta räknas ut som: $T_n = \frac{n(n+1)}{2}$

Med andra ord kan vi när $n = 3$ beräkna triangeltalet som: $\frac{3(3+1)}{2} = 6$

Din uppgift är att hitta alla triangeltal mellan 1 och 500 där det också finns ett heltal mellan 1 och 500 som är en jämn kvadratrot av det triangeltalet. Alltså att det finns ett heltal n (*ett triangeltal mellan 1 och 500*) och k (*ett heltal mellan 1 och 500*) där följande är sant:

$$T_n = k^2 \;\; \textit{vilket kan skrivas som} \;\; \frac{n(n+1)}{2} = k^2$$

Dessa tal ska du hitta med hjälp av `uppgift5.rb` ! Modifiera `my_problem` för att testa alla n och k och räkna ut triangeltalet. Därefter behöver du kontrollera om k^2 är lika med triangeltalet. Om de är lika så skriver du ut båda talen.

Hur denna utskrift görs spelar ingen roll, så det behöver inte matcha exemplet nedan exakt. Däremot måste det framgå vilket tal som är n och k respektive, samt när alla möjligheter har testats.

Funktionella krav:

```
n = 1, k = 1
n = 8, k = 6
n = 49, k = 35
n = 288, k = 204
Det var alla möjligheter.
```

Matematiskt går det att uttrycka denna uppgift väldigt kompakt. Som en rolig grej tillhandahåller vi därför en matematisk beskrivning av problemet ovan som en liten primer för kommande mattekurs:

Använd AMB-klassen för att skriva ut $\forall n, k$ så att: $(0 < n \leq 500) \wedge (0 < k \leq 500) \wedge (\frac{n(n+1)}{2} = k^2)$

Del 3: Uppgift 6 - XML-parsning

I den givna filen `large_shipments.xml` finner du en lista med paket som ska levereras. Det viktiga att hålla koll på för varje paket är **paketets ID** (*det `id` som finns för varje `package`-tagg*), **paketets vikt** (*taggen `weight` för varje paket*), **var paketet ska levereras** (*taggen `destination` för varje paket*) samt om leveransen är av typen **Express eller Economy**.

Målet med uppgiften är att med en SAX- eller DOM-parser läsa ut information från xml-filen och fylla tre lastbilar med lämpliga paket. Lastbilarna går till Linköping, Stockholm, och Göteborg. Varje lastbil kan frakta upp till 1500 kg.

Varje lastbil ska lastas med så många paket som möjligt (max 1500 kg) som ska till samma plats som lastbilen. Dessutom så har företaget en liten backlog, så ledningen har beslutat att endast frakta de paket som är av typen `Express`. Du behöver inte tänka på att optimera lasten av bilarna, stoppa i paket i varje lastbil tills det inte längre går bara.

Du behöver lagra vilka paket-IDn som lastats på varje lastbil, hur mycket alla dessa paket väger tillsammans, samt vilken destination paketen ska till. Detta skrivs sedan ut i terminalen. Din utskrift behöver inte matcha den nedan, men det behöver vara tydligt vilken destination dessa paket ska till, hur mycket totalvikten är samt vilka ID som hör till vilken lastbil.

Funktionella krav:

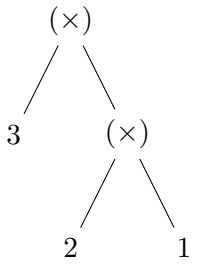
```
Stockholm:
  Packages: P13, P177, P216, P122
  Total weight: 1247 kg

Linköping:
  Packages: P99, P204, P26, P24
  Total weight: 1290 kg

Göteborg:
  Packages: P59, P93, P2, P90, P221, P223
  Total weight: 1455 kg
```

Del 3: Uppgift 7 - Parser AST

I den givna filen `uppgift7.rb` finner du den parser vi är vana vid från labbserien. I denna parser finns en förberedd `FactorialParser` som du ska bygga klart.

Denna parser behöver kunna hantera addition, subtraktion, fakultet, samt heltal. Som en påminnelse så är fakulteten av ett heltal produkten av alla tal från 1 till och med talet själv, och skrivs med ett utropstecken efter talet. Alltså, fakulteten av 3 skrivs som (och blir) $3! = 3 \times 2 \times 1$, och likadant för fakulteten av 5 som blir $5! = 5 \times 4 \times 3 \times 2 \times 1$.

Förutom att skriva om parsen så att den kan hantera aritmetiken så ska du också se till att parsern nu returnerar ett abstrakt syntaxträd istället för att direkt utvärdera uttrycken. För full poäng ska desutom delen av trädet som representerar fakulteten byggas upp som flera multiplikationer. Nedan följer ett exempel av hur ett delträd som representerar 3! kan se ut:



Programmet behöver endast kunna hantera positiva heltal. Efter du konstruerat syntaxträden ska de utvärderas och resultatet av utvärderingen ska skrivas ut. Exakt hur dessa utskrifter ser ut spelar ingen roll, så de behöver inte matcha exemplet nedan – men däremot bör det framgå vilket test som körs och vad resultatet blir (vilket bör vara korrekt).

Funktionella krav:

```
4! + 5! = 144
3! + 2! = 8
7! - 6! = 4320
25 - 4! = 1
5! + (4! - 3!) = 138
```