TDP007 - Tenta

2020-06-10

Regler

- All kod efterrättas på denna tenta
- 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 via Zoom
- 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.

Tentan består av uppgifter (några indelade i deluppgifter) som totalt kan ge 50 poäng.

Ruby-docs

På tentan har du tillgång till referenssidorna på `https://ruby-doc.org/` (både core och std-lib finns tillgänglig) via webbläsaren.

Rubular

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

Givna filer

Eventuella givna filer finns länkade från kurssidan.

Avslutning

När du skickat in alla uppgifter och känner dig färdig kan du stänga tetaklienten, lämna Zoom och stänga thinlinc.

Uppgift 1 - Regexp (7+5p)

Praktisk (7p)

I den givna filen `pagefoot.html` finns sidfooten från kurssidan. Denna sidfot innehåller många länkar som ligger i html-element av typen `<a>`. Ditt jobb är att läsa in filen och med Regexp identifiera länkarna samt det som skrivs ut i webbläsaren. Vi vill alltså komma åt det som står efter likhetstecknet efter `href` samt innehållet i själva elementet. I html-elementet `<a href="enlänk"> ettinnehåll </a>` skulle innehållet vara 'ettinnehåll' och länken skulle vara 'enlänk'.

När du identifierat alla dessa element ska du skriva ut antalet du hittat (bör vara 27) samt spara undan denna information i en `Hash` där innehållet är nyckeln och länken värdet. Du ser säkert att vissa länkar inte ser giltiga ut, exempelvis `/department/contact/index.sv.shtml`. Detta beror på att de är dynamiska och använder kurssidans grund som utgångspunkt. Du skall lägga till `https://ida.liu.se` i början av dessa. När du är färdig bör resultatet av körningen av programmet se ut på följande sätt:

```
Number of links found: 27
{"Kontakta IDA"=>"https://ida.liu.se/department/contact/index.sv.shtml",
 "Kartor"=>"https://ida.liu.se/department/location/index.sv.shtml",
 "Om webbplatsen"=>"https://liu.se/om-webbplatsen",
  .   .   .
 "Internwebb"=>"https://ida.liu.se//local/ida/index.sv.shtml",
 "Akut"=>"https://ida.liu.se/department/emergency/index.sv.shtml"}
```

Formateringen är inte viktig vid utskriften men jag rekommenderar att du skriver ut `Hashen` på följande sätt för läsbarhetens skull:

```
require 'pp'
pp my_hash
```

Ickefunktionella krav för full poäng:

- Regepx som hittar alla relevanta html-element
- Matchgroups ska användas för att hämta ut innehållet och länken ur elementet

Teoretisk (5p)

Besvara dessa frågor i en separat textfil och lämna in på samma uppgift.

- Konstruera ett reguljärt uttryck som kan matcha en korrekt e-postadress. Förklara vilka hänsyns taganden du gjorde och vad varje del i ditt regexp faktiskt gör.
- Hitta på ett kodexempel (som inte är taget ur boken) som på ett bra sätt visar hur man kan använda iteratorer. Skriv sedan samma kod, men helt utan att använda iteratorer.

Uppgift 2 - XML (9+4p)

Praktisk (9p)

I filen `adventure.xml` hittar du ett antal platser som ingår i ett litet textbaserat äventyrsspel. Varje plats har ett `id` som är unikt och en beskrivning. Utöver det har också platserna anslutningar till andra platser. Varje anslutning har ett väderstreck och det `id` vars plats man kommer till om man färdas i det väderstrecket.

Ditt jobb är att skriva ett program som låter användaren färdas mellan olika platser genom att mata in väderstreck. När användaren kommer till en plats ska platsens beskrivning skrivas ut enligt formatet i körexemplet. Sedan ska alla anslutningarna skrivas ut enligt körexemplet. Efter det får användaren frågan om vart de vill gå nu och matar sedan in ett väderstreck.

I denna uppgift kan vi anta att användaren alltid matar in ett giltigt väderstreck för platsen de är på just nu. Användaren börjar alltid sin resa vid vattendraget.

Körexempel:

```
You see a: A babbling brooke
To the west you see a path to the clearing
Which way do you go?: west

You see a: A lush forrest clearing
To the east you see a path to the riverbank
To the north you see a path to the cave
Which way do you go?: north

You see a: A dark cave, with a light at the end
To the south you see a path to the clearing
To the west you see a path to the town
Which way do you go?: west

You see a: A nice little town
To the east you see a path to the cave
Which way do you go?: east

You see a: A dark cave, with a light at the end
To the south you see a path to the clearing
To the west you see a path to the town
Which way do you go?:
```

Teoretisk (4p)

Du löste den ovanstående uppgiften antingen med en SAX- eller DOM-parser. Beskriv hur du skulle löst uppgiften med den andra tekniken. Ditt jobb här är att övertyga oss om att du hade kunnat lösa problemet med den andra tekniken.

Uppgift 3 - parser (9+4p)

Praktisk (9p)

Smörängens skola vill ha ett program som introducerar barn till att skriva och prata om matematik. Det vill att det ska vara på svenska men har bara tillgång till amerikanska tangentbord. Alla å, ä och ö blir därför a respektive o i detta språk. Ditt jobb är att modifiera parserna i rdparse.rb så att språket som följande grammatik definierar fungerar:

```
VALID ::=  EXPR | COMP

EXPR ::=   NUMBER
           | EXPR plus EXPR
           | EXPR minus EXPR

COMP ::=   EXPR is EXPR
           | EXPR ar mindre an EXPR
           | EXPR ar storre an EXPR

NUMBER ::= NUM | negativ NUM

NUM ::=    ett | tva | tre | ... | tio
```

Resultaten av en beräkning ska vara ett tal och resultatet av en jämförelse ska vara antingen ja eller nej. Man kan använda negativa tal genom att skriva negativ framför. Vad som händer om man råkar gå utanför intervallet -10 till +10 är odefinierat, d.v.s. det spelar ingen roll.

Omvandlingen mellan tal som strängar ('ett, 'tva', etc) och siffror (1, 2, etc) gör du lämpligen inte i parserreglerna. Det är dock helt okej att ha en hårdkodad tabell.

Körexempel:

```
[NumberParser] ett plus tre
=> 4
[NumberParser] tva plus tre ar fem
=> ja
[NumberParser] fem ar mindre an fyra plus tva
=> ja
[NumberParser] fem plus tio ar storre an fyra plus tva
=> ja
```

Teoretisk (4p)

Det språk du nu implementerat tillåter en mycket begränsad del av aritmetik. Förklara hur du kan utöka språket för att tillåta definition av variabler

Uppgift 4 - DSL (8+4p)

Praktisk (9p)

I files `dsl.rb` finns en kvittofil. Du ska implementera ett litet DSL för att hantera denna sorts filer. Filen är strukturerad så att man inleder ett kvitto med `reciept "namn på konfigurationen"`. Allting som kommer under den raden fram till `end reciept` eventuellt förekommer igen tillhör det kvittot. I `dsl.rb` finns 2 kvitton. Du ska med hjälp av method_missing skapa en klass. Denna klass ska ha en funktion `read` som tar in ett filnamn och ett namn på en konfiguration. funktionen ska sedan läsa in och spara all data som hör till den konfigurationen i en hash(se körexemplet). I den givna filen `uppgift4.rb` finns en påbörjad lösning på problemet.

Det är ok att lägga till andra medlemsfunktioner för att lösa problemet men man ska kunna stoppa in godtyckliga nyckel-värde par i konfigurationsfilen.

Körexempel:

```
100032:
{:store_name=>"pontus bokhandel",
 :company_name=>"haglunds bokhandel ab",
 :first_name=>"pontus",
 :last_name=>"haglund"}

100033:
{:store_name=>"alexandras bokhandel",
 :first_name=>"alexandra",
 :last_name=>"berggren"}
```

Teoretisk (4p)

1. I denna uppgift har du använt dig av en klassfunktion som heter `read` för att lösa problemet. Denna typ av funktion är också kända som statiska funktioner i exempelvis c++. Redogör för hur en klassfunktion skiljer sig från en medlemsfunktion
2. Vad är ett domänspecifikt språk? Hur skiljer sig det från andra språk du stött på? Var går gränsen mellan dsl och gpl?