TDP007 - Tenta

2024-08-20

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 jourhavande 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.

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 och logga ut.

Uppgift 1 - Teori och grundläggande ruby (10p)

Uppgift 1a (10p)

Du har fått i uppgift att designa ett python-liknande språk som en uppgift under en arbetsintervju. Du behöver inte skapa en implementation av språket i denna uppgift men visa på att du kan hantera de underliggande principerna, verktyget som valts för språket är rdparse (som du arbetat med i kursen, vilken tur). Med hjälp av rdparse ska du ange signaturerna för reglerna och tokeniseringen som krävs för att uppfylla språkets krav, kraven beskrivs längre ner i uppgiften. Ett exmpel på hur ett svar skulle kunna se ut för diceroller språket skulle vara (plus text som förklarar sådant som inte är självklart från reglerna, exempelvis variabellagring eller hur tärningen rullas):

```
token(/\s+/)
token(/\d+/)
token(/./)

start :expr do
  match(:expr, '+', :term)
  match(:expr, '-', :term)
  match(:term)
end

rule :term do
  match(:term, '*', :dice)
  match(:term, '/', :dice)
  match(:dice)
end

rule :dice do
  match(:atom, 'd', :sides)
  match('d', :sides)
  match(:atom)
end

rule :sides do
  match('%')
  match(:atom)

rule :atom do
  match(Integer)
  match('(', :expr, ')')
```

Uppgiften definerar följande features i språket:

- Heltal och strängar som datatyper
- Aritmetik med addition och multiplikation med tillhörande parenteser
- Tilldelning av variabler
- Uppslagning (användning av sparade värden) av variabler
- Utskrift

Uppgift 2 - rdparse lägg till AST(10p)

I den givna filen `rdparse.rb` finns tärningsprogrammet som vi arbetade med under kursen. Ditt jobb är att ändra den givna parserna så att den genererar ett abstrakt syntaxträd (AST) istället för att direkt utvärdera uttrycken. Syntaxträdet skall sedan utvärderas när **alla** noderna i trädet skapats. Utvärderingen kan ske inuti en **lämplig** regel eller i funktionen `parse`.

Uppgift 3 - DOM-Parsning av XML(10p)

I den givna filen `world.xml` finns världen till ett enkelt textbaserat spel. I denna typ av spel får man uppleva spelvärlden genom text och interagerar genom att skriva var man vill gå. Varje plats i spelet finns i `world.xml` och varje plats har en beskrivning och ett antal anslutande platser som man kan ta sig till. Dessa associeras med andra platser genom ett unikt id som finns i attributet `target`. Ditt jobb är att med hjälp av en dom-parser implementera programmet som låter användaren spela spelet och gå runt i världen. Man börjar på platsen som har id 1. Om man sedan skriver `east` så kommer man komma till platsen som har id 3, och så vidare. Se körexemplet:

Körexempel:

```
Welcome to the Adventure Game!

Forest
You are in a dark forest. The trees tower above you, and you hear...

You can go:
 - north
 - east

What do you do? east

River
A wide river flows swiftly here. The current is strong, and the water is...

You can go:
 - west
 - east

What do you do? east

Village
You find yourself in a small village. Smoke rises from the chimneys...

You can go:
 - west
 - north

What do you do?
```

Utskriften trunkeras i körexemplet på grund av radlängden, du behöver inte göra det i din lösning.

Krav: Din lösning behöver vara robust nog för att hantera annorlunda ordning på elementen i xml-filen. Inbördes ordning av elementen får inte spela roll. Programmet behöver så klart fungera oavsett innehållet i xml-filen (så länge den har samma format) och oavsett vilken ordning spelaren besöker platserna.

Uppgift 4 - SAX-Parsning av XML(11p)

Se uppgift 3. I denna uppgift presenteras samma problem men din lösning skall istället använda en sax-parser. Det finns inga krav på prestanda i denna uppgift, du kan antingen söka igenom hela filen för varje steg eller spara undan dessa i en lämplig datastruktur och sedan arbeta med den.

Uppgift 5 - dsl (9p)

I den givna filen `dsl.rb` finns ett givet program skrivet i ett domänspecifikt språk, programmet är en beskrivning av ett brädspel. Justera den givna koden i `dsl_reader.rb` så att innehållet i filen läses in och sparas i en hash, se körexemplet. Datatyperna är viktiga i det här sammanhanget och ska matcha körexemplet.

Varje rad i `dsl.rb` associerar data med en nyckel. Först skrivs nyckeln följt av en lista av data som ska associeras med det namnet. Datan seppareras med kommatecken. Om bara en data ges efter nyckeln ska detta inte sparas som en lista, se körexemplet där name associerad med en sträng, men genre associeras med en lista av strängar.

Lägg till en lämplig print-funktion i din dsl-läsare och tillhörande anrop i huvudprogrammet så programmet skriver ut resultatet med lämpligt format (formateringen av utskriften behöver inte följa körexemplet).

Körexempel:

```
{"name"=>"Descent: Journeys in the Dark second edition",
 "min_age"=>14,
 "time"=>120,
 "genre"=>["action", "adventure", "dungeon"]}
```