TDP007 - Dugga 2

2023-03-01

Regler

- All kod efterrättas på denna dugga
- Inget sent insläpp tillåts under duggan
- 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 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)

1. Vad är en lexer och hur fungerar den i grova drag? (1p)
2. Vad är ett domänspecifikt språk? Beskriv kortfattat en typisk situation där man kan vilja använda ett domänspecifikt språk och motivera varför. (2p)
3. Beskriv åtminstone två olika faktorer som kan bidra till att ett datorspråk blir populärt och motivera varför. (2p)

Operator	Prioritet	Beskrivning
+	1	Addition
-	1	Subtraktion
*	2	Multiplikation
/	2	Division
^	3	Potens
%	4	Modulo (rest)

Uppgift 2 - Parser (7p)

I denna uppgift ska du använda parsern i den givna filen rdparse.rb. Du kan modifiera det existerande diceroller-språket eller skapa ett nytt. Du ska skapa ett verktyg för att utvärdera aritmetiska uttryck. Men dessa aritmetiska uttryck skall utvärderas på ett sätt som är oväntat för en ovetande slutanvändare. I språket skall operatorerna i tabellen finnas och fungera för godtyckliga sammansatta uttryck, addition och subtraktion har alltså delad högst prioritet i detta språk och modulo har lägst prioritet.

Det finns inte heller några parenteser i detta språk för att göra saker extra förvirrande. Trots att prioriteten är konstig i detta språk så ska associativiteten fungera som man vanligtvis förväntar sig.

Körexempel:

```
[aritparser] 1 + 2 * 3
=> 9

[aritparser] 2 ^ 7 % 4
=> 0

[aritparser] 2 ^ 3 ^ 2
=> 512
```

Uppgift 3 - Ickedetermenistisk problemlösning (7p)

I filen `amb_test.rb` finns den problemlösare vi arbetade med under föreläsningen. Nedanstående figur motsvarar placeringen av 4 datorer i en labbsal (A, B, C, D) samt 4 eluttag i samma sal (0, 1, 2, 3).

```
. 0 . . 1
. . . D .
. B . A 2
. . . . .
C . 3 . .
```

Varje eluttag kan bara stödja en inkopplad dator. Den längsta kabeln som it-avdelningen har tillgång till är 3 enheter lång och det finns inga skarvsladdar. Ditt jobb är att ta reda på alla kombinationer som man kan koppla datorerna till eluttagen. Kabeln kan enbart dras ortagonalt, avståndet mellan dator D och uttag 1 är alltså 2. Här är ett exempel på hur datorerna och avstånden kan representeras:

```
a = [[0, 4], [1, 3], [2, 1], [3, 3]]
b = [[0, 2], [1, 5], [2, 3], [3, 3]]
c = [[0, 5], [1, 8], [2, 6], [3, 2]]
d = [[0, 3], [1, 2], [2, 2], [3, 4]]
```

Där d1 är dator 1 som har 4 kopplingsmöjligheter. Avståndet till uttag 0 är 4, avståndet till uttag 1 är 3 och så vidare.

Körexempel:

```
Dator 1 - Uttag 1, Dator 2 - Uttag 0, Dator 3 - Uttag 3, Dator 4 - Uttag 2
Dator 1 - Uttag 1, Dator 2 - Uttag 2, Dator 3 - Uttag 3, Dator 4 - Uttag 0
Dator 1 - Uttag 2, Dator 2 - Uttag 0, Dator 3 - Uttag 3, Dator 4 - Uttag 1
Det var alla möjligheter
```

Tips:

- Metoden `choose` i problemlösaren arbetar i det givna exemplet med ett intervall. Du kan använda flera sådana intervall för att testa olika index i din valda datarepresentation.
- Det första som behöver testas är att ingen kabel är längre än den givna maximala längden
- Det andra som behöver testas är att ingen dator är kopplad till samma uttag.
- Även om problemet i huvudsak skall lösas med problemlösaren så är det ok att lösa delproblem, så som att kontrollera om en Array endast innehåller unika data, med vanlig imperativ problemlösning

Uppgift 4 - Constraint propagation networks(6p)

I filen `constraint_networks.rb` finns de restriktionsnät som vi arbetat med under föreläsningar och seminarier. Din uppgift är att konstruera en ny constraint som kan slå ihop två `Array`er till en. Vi tänker oss att denna nya constraint tar två indata-Arrayer, **a** och **b**, och som resultat ger en utdata-Array **c** som är en sammanslagning av dessa två Arrayer. Om vi har skapat tre `Connectors` och satt ihop dessa med hjälp av vår nya constraint borde vi kunna skriva så här:

```
a.user_assign([1, 3, 5])
b.user_assign([2, 4, 6])
c.value == [1, 3, 5, 2, 4, 6]
```

Vår nya constraint ska fungera åt båda hållen. Om vi t.ex. känner till värdet på a och c bör vi kunna räkna ut värdet på b. Du behöver inte nödvändigtvis lösa problemen med de tidigare existerande constraintsen.