TDP007 - Dugga 2

2020-03-10

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 duggans gång är strängt förbjuden.
- Böcker och anteckningssidor kan komma att granskas av assistent, vakt eller examinator under duggans gång.
- Frågor om specifika uppgifter eller om duggan i stort ska ställas via tentasystemets kommunikationsklient.
- Systemfrågor kan ställas till assistent i sal genom att räcka upp handen.
- Endast uppgifter inskickade före dugganstidens 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 duggan

Duggan 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 - Teorifrågor (7p)

1. (1p) Förklara hur en parser skiljer sig från en lexer i grova drag.
2. (2p) Välj ett programspråk som du är bekant med och som är någorlunda välkänt. Argumentera för hurvida detta språk är eller inte är domänspecifikt.
3. (2p) Sedan ni började studera inom IP har vi arbetat med strukturerad programmering och hållt oss borta från funktionalitet så som GO-TO. I denna kurs har vi dock använt kod som är byggd runt continuations i Ruby. Argumentera för när det kan vara rimligt att använda sådan funktionalitet och varför.
4. (2p) Java, python och c/c++ är mycket populära språk, men varför? Det finns en uppsjö av olika programspråk som sällan används trots att de är sunda språk. Motivera varför vissa programspråk blir populära och andra inte.

Uppgift 2 - DSL (6p)

Janeway gillar att spela Dungeons & Dragons med sina vänner. Tyvärr har hennes vänner ett liv och kan inte spela alla vardagar i veckorna. Hon har därför tvingat dem att fylla i sitt namn alla dagar de kan spela under de kommande veckorna.

I denna uppgift ska du implementera tolken för ett `dsl` som du hittar i `friends.rb`. Språket är byggt så att man anger en vecka och sedan anger man dagarna som finns under den veckan. Efter varje dag står namnet på de vänner som kan närvara under den dagen.

Utgå ifrån filen `dsl_reader.rb` och implementera det som kvarstår så att programmet går att köra enligt körexemplet nedan. Den statiska funktionen (klass funktionen) `avail` anropas i exemplet nedan med filnamnet och namnet på kompisen `Christine`.

För full poäng bör åtminstone veckodagarna läsas med hjälp av `method_missing`. Körningen ska också se ut som körexemplet.

```
MeetupReader.avail("friends.rb", "Christine")
```

```
Christine is available the following days:
week34: monday wednesday thursday friday
week35: wednesday thursday friday
```

Uppgift 3 - Ickedeterministisk programmering (6p)

Rollspelet Dungeons & Dragons är det mest populära bordsrollspelet i världen. I filen `dnd_inventory` finns en array med produkter. Varje produkt representeras av en `hash` och har ett namn och ett pris. Ditt jobb är att beskriva följande problem och lösa det med hjälp av problemlösaren som finns implementerad i `amb_test.rb`. Det är ok att kopiera in hashen från `dnd_inventory.rb` i problemlösaren eller huvudprogrammet.

Ditt program ska plocka ut alla kombinationer av 4 produkter som uppfyller följande krav:

- Första produkten måste ha namnet `Players Handbook`
- Ingen produkt får förekomma mer än en gång
- Produkterna får sammanlagt inte kosta mer än 890 kr

Skriv ut de kombinationerna som uppfyller kraven och antalet kombinationer du hittar.

Uppgift 4 - Rdparse (6p)

Du ska i denna uppgift skapa ett program som kan läsa in en kortlek. Du ska göra detta genom att utöka rdparse som du hittar i filen `rdparse.rb`. En påbörjad lösning finns i filen `deck_parser.rb` När parsern startar ska användaren kunna skriva in spelkort i parsern på formatet `[suit][value]` eller `[value][suit]`. Parsern ska sedan returnera en lista av alla korten som lagts till och skriva ut listan. Se körexemplet.

Utskriften behöver inte se ut exakt som i exemplet men funktionaliteten ska stämma. Kortleken ska alltså fyllas på efter användaren matar in ett eller fler kort.

`[suit]` kan vara hjärter eller spader och anges med bokstäverna `h` och `s`. `value` kan vara alla vanliga valörer och anges med `1,2,3,4,5,6,7,8,9,10,j,q,k`

```
[deck specifier] 9h kh qs
Suit: hearts, Value: 9
Suit: hearts, Value: king
Suit: spades, Value: queen
[deck specifier] 10s
Suit: hearts, Value: 9
Suit: hearts, Value: king
Suit: spades, Value: queen
Suit: spades, Value: 10
[deck specifier]
```