Kontrollskrivning i Programmeringsparadigm - DD1361 och DD1362 2019-03-11 08.00-10.00

Inga hjälpmedel är tillåtna. Skriv inga svar på blanketten. För godkänt på en del krävs utförliga svar med kompletta meningar på samtliga (del)uppgifter för den delen. Ett grovt fel ger underkänt. En del blir godkänd även med ett halvgrovt fel. På den funktionella delen är det även tillåtet att göra ett halvgrovt fel för var och en av bonusdatumen på F1 och F2 som uppfyllts. Om det finns oredovisade labbar som lämnades in innan bonusdatum och dessa påverkar KS:ens godkäntstatus så ges betyg Fx och labbredovisningen räknas som komplettering. Lycka till önskar Marcus och Per!

Del 1: Funktionell programmering

1. Förklara med varsitt exempel skillnaden mellan överlagring (overload) och överskuggning (override). Exemplen behöver inte vara skrivna i Haskell.

2. En Haskellfunktion har typsignaturen:

   ```
   primes::[Integer]
   ```

   Den representerar en oändlig lista med primtal. Några påståenden om denna funktion:

   (a) Anropet

   ```
   primes 42
   ```

   representerar de första 42 primtalen.

   (b) Om primes anropas från ghci med

   ```
   Prelude> primes
   ```

   så kommer den att fastna i en oändlig loop.

   (c) Haskell har en egenskap som gör det möjligt att anropa primes utan att det blir en oändlig loop.

   Vilket eller vilka av dessa påståenden stämmer? Motivera svaret utförligt med kursens terminologi.

3. I Haskell finns funktionerna map och foldl. Dessa båda är exempel på funktioner som tar en eller flera funktioner som parametrar. Funktioner som gör detta (eller returnerar en funktion) har ett samlingsnamn.

   (a) Vad är detta samlingsnamn?

   (b) Vad är det för skillnad i tillämpningen av funktionen vid map respektive foldl?

4. En Haskellfunktion har typsignaturen

   ```
   combine :: String -> String -> String
   ```

   (a) Hur många parametrar kan man som mest anropa funktionen med?

   (b) Vad returnerar funktionen om man anropar den med färre parametrar än i föregående uppgift och vad kallas principen?

Del 2: Programmeringsparadigm

Paradigmdelen gäller endast studenter som är registrerade på DD1362 (årets kurs) och inte studenter som vill komplettera DD1361 (den gamla programmeringsparadigmkursen).

5. I imperativ programmering finns en form av subprogram som i fallet assembler kallas subrutin. I procedurell programmering finns en form av subprogram som kallas procedur. I var och en av dessa paradigm förekommer följande typer av variabler och värden.

   (a) Parametervärden (kallas också argument).

   (b) Variabler som deklareras i funktionen/subrutinen/proceduren/subprogrammet.

   (c) Returvärden.

   Redogör för vilka villkor och möjligheter det imperativa respektive det procedurella paradigmet ställer upp för att läsa och uppdatera dessa variabler och värden.

6. Lambdakalkylen är uppbyggd kring tre regler som beskriver hur lambdatermer (eller lambdauttryck) kan konstrueras. Den första regeln säger att det finns variabler och konstanter. Vad heter de andra två reglerna och hur kan dessa se ut när de tillämpas? Visa reglerna matematiskt eller i valfritt programmeringsspråk. Om du väljer ett programmeringsspråk, ange vilket språk som du har använt.

7. På sista föreläsningen i paradigm nämndes strömmar i Java (Streams). Dessa infördes i Java 1.8 trots att de tillhör ett annat paradigm än de som användes till och med Java 1.7. Namnge strömmarnas paradigm. Redogör också för varför strömmar inte tillhör det imperativa paradigmet.