Tenta i Programmeringsparadigm - DD1362 2019-06-01 09.00-14.00

Inga hjälpmedel är tillåtna. Skriv inga svar på blanketten. Ett grovt fel ger underkänt på en del. Ett halvgrovt fel per del är ok. Bonuspoäng från Inet-labben våren 2019 ger utrymme för två halvgrova fel (men inte ett grovt fel) på en av delarna. **Du behöver inte skriva de delar där du redan är godkänd på kontrollskrivningen eller tidigare års tentor.**

Del 1: Funktionell programmering

1. Ellinor och Alexandra programmerar en Haskellfunktion för beräkning av triangeltal. Ellinor menar att basfallet ska implementeras såhär:

```
h :: Int -> Int
h 0 = 0
h n = n + (h (n-1))
```

   Alexandra tycker istället att den ska implementeras såhär:

```
h :: Int -> Int
h n
    |n==0 = n
    |otherwise = n + (h (n-1))
```

   (A) Vad kallas tekniken för att implementera basfallet i Ellinors lösning?

   (B) Vad kallas tekniken för att implementera basfallet i Alexandras lösning?

   (C) Svara på och motivera om någon av Ellinors eller Alexandras lösning är svansrekursiv.

2. Studera följande typsignatur:
   `pickles :: [Int]`
   Beskriv i detalj med kursens terminologi vad som händer vid anropet:
   `take 9 pickles`.

3. Studera följande typsignatur:
   `g :: Int -> Int -> Int`
   Beskriv i detalj med kursens terminologi vad som händer vid följande anrop:

   (A) `g 7`

   (B) `g 17 42`

   (C) `g 17 43 109`

4. Studera följande typsignatur:
   `f :: (Int -> Int) -> [Int] -> [Int]`
   Beskriv i detalj med kursens terminologi vad vi vet om `f`.

Del 2: Formella språk och syntaxanalys

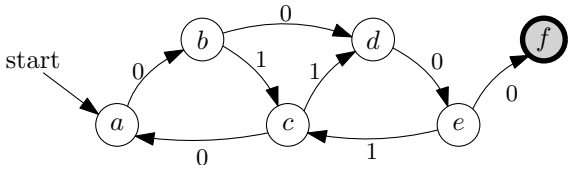
5. Förklara följande terminologi från kursavsnittet. Förklara i vilket sammanhang begreppet dyker upp, vad det betyder, och **ge ett konkret exempel** där du använder begreppet i en situation på ett korrekt sätt.

   (a) "Icke-slutsymbol".

   (b) "Lexikal analys".

   (c) "Reguljärt språk".

6. Betrakta följande ändliga automat (DFA). Det gråmarkerade tillståndet är accepterande, de andra inte.



   (a) Vilka tecken kan förekomma i de strängar som automaten accepterar?

   (b) Ge **tre** exempel på strängar som *inte accepteras* av automaten. Strängarna får *endast* använda sig av de tecken som ingår i svaret på (a).

   (c) Ge **tre** exempel på strängar som accepteras av automaten. För *ett* av dina tre exempel, beskriv steg för steg hur automaten exekveras på strängen.

   (d) Är språket som automaten beskriver ändligt? (Svara "ja" eller "nej", inte något annat!)

7. Betrakta följande grammatik. Startsymbol är `<Plot>`.

```
<Plot>  ::= 'f' <Fire> | 'i' <Ice>
<Fire>  ::= 'f' <Ice>  | 'i'        | 'w' <Plot> <Ice>
<Ice>   ::= 'f'        | 'i' <Fire> | 'w' <Plot> <Fire>
```

   Nedan finns pseudo-kod för en rekursiv medåknings-parser för denna grammatik. Du kan anta att funktionen `nextToken()` hämtar nästa tecken i indatasträngen.

   (a) Beskriv tydligt och i detalj hur parsern kommer exekvera steg för steg på indatasträngen "`iwifi`". Visa vilka funktions-anrop som görs, vilka val som tas i olika if-satser, och vilken nivå av rekursionen exekveringen befinner sig i (tips: rekursionen visas enklast genom indentering).

   (b) Ge ett exempel på en sträng som *börjar* på "`ff`" *och* där körning av parsern kommer att resultera i att SyntaxError kastas på **rad 19** i koden.

```
1  function Plot():
2     x = nextToken()
3     if x == 'f':
4         Fire()
5     else if x == 'i':
6         Ice()
7     else:
8         throw exception SyntaxError()
9
10 function Fire():
11     x = nextToken()
12     if x == 'f':
13         Ice()
```

```
14      else if x == 'i':
15          // nothing to do here
16      else if x == 'w':
17          Plot()
18          Ice()
19      else:
20          throw exception SyntaxError()
21
22  function Ice():
23      x = nextToken()
24      if x == 'f':
25          // nothing to do here
26      else if x == 'i':
27          Fire()
28      else if x == 'w':
29          Plot()
30          Fire()
31      else:
32          throw exception SyntaxError()
```

Del 3: Programmeringsparadigm

8. Vilket krav ställer det procedurella paradigmet på funktioner (subprogram) som inte ställs av det strukturerade paradigmet? Redogör i detalj med kursens terminologi.

9. Vi har pratat om högnivåspråk och lågnivåspråk i kursen. Här är några språk som är sorterade på antalet bokstäver i språkets namn. Sortera dem från lågnivå till högnivå och motivera för vart och ett av dem varför de hamnade på sin plats:
   `[C, Java, Haskell, Assembler]`.
   Det är tillåtet att byta ut Java mot Python och sortera den listan istället. Ha språket som är mest lågnivå till vänster och mest högnivå till höger. Förklara också vad vi i kursen menar med lågnivåspråk.

10. Nämn två egenskaper som programspråk som tillhör det objektorienterade paradigmet har som vi inte kan förutsätta finns i ett språk som endast tillhör det imperativa paradigmet. Ge exempel på ett programspråk som är objektorienterat. Ge också ett exempel på ett programspråk som är imperativt men inte objektorienterat.

11. Skissa en datormodell enligt Von Neumann-arkitekturen (rita figur) och beskriv var flaskhalsen i modellen ligger.