Tentamen (del 1) (6 högskolepoäng) i Programkonstruktion och datastrukturer (1DL201)

Pierre Flener

Onsdag 21 december 2011, kl. 08:00 – 11:00, i Polacksbacken skrivsal

Hjälpmedel: Inga. ***Inte*** heller elektronisk utrustning.

Hjälp: En av huvudlärarna kommer att besöka skrivsalen c:a klockan 09:00.

Anvisningar: Markera i tabellen nedan ***inte mer än ett*** svar per fråga genom att kryssa över bokstaven för det svarsalternativ som du väljer. ***Lämna bara in denna sida.*** Det är inte meningen att du skall lämna kommentarer till dina svar. Om du tycker att någon fråga är oklar eller felaktig, markera fråganumret med en $\star$ på ***den här*** sidan, och förklara ***på baksidan av detta blad*** vad du menar att problemet är och vilka antaganden du gjort för att kunna svara på frågan.

	Fråga	Svar					Fråga	Svar				
betyg **3 U**	1	A	B	C	D	E	2	A	B	C	D	E
	3	A	B	C	D	E	4	A	B	C	D	E
	5	A	B	C	D	E	6	A	B	C	D	E
	7	A	B	C	D	E	8	A	B	C	D	E
	9	A	B	C	D	E	10	A	B	C	D	E
betyg **4**	11	A	B	C	D	E	12	A	B	C	D	E
	13	A	B	C	D	E	14	A	B	C	D	E
	15	A	B	C	D	E						
betyg **5**							16	A	B	C	D	E
	17	A	B	C	D	E	18	A	B	C	D	E
	19	A	B	C	D	E	20	A	B	C	D	E

Identitet: Din tentakod (eller namn och personnummer om du saknar kod):

. . . . . .

Frågeunderlag

En del av frågorna kommer att behandla programmet `listOfWords` som skall ta en sträng och skapa en lista av alla ord i strängen. Med ord menas här sammanhängande följder av tecken som inte är blanka. Till exempel så skall alltså anropet
`listOfWords "Snart är det sommar."` ge listan `["Snart","är","det","sommar."]`.

Funktionsspecifikation och programkod för `listOfWords` är:

`listOfWords text`
TYPE: `string -> string list`
PRE: Ingen.
POST: En lista av orden i text. (Ord skiljs åt av blanktecken.)
EXAMPLE: `listOfWords "Snart är  det sommar." = ["Snart","är","det","sommar."]`

```
fun listOfWords text = listOfWords'(text,0)
```

Nedan finns funktionsspecifikation och programkod för hjälpfunktionen `listOfWords'`. Båda är ofullständiga – det finns luckor i dem, markerade `?1?`, `?2?` etc. Din uppgift blir att tala om vad som skall finnas i luckorna.

`listOfWords'(text,pos)`
TYPE: `?1?`
PRE: `?2?`
POST: En lista av orden i text, från och med position pos
EXAMPLE: `listOfWords'("Snart är  det sommar.",0) = ["Snart","är","det","sommar."]`
VARIANT: `?6?`

```
fun listOfWords'(text,pos) =
   if pos = size text then
     []
   else
     let
       val firstspace = findSpace(text,pos)
      in
        if ?3? then
          listOfWords'(text,pos+1)
        else
          ?4? :: ?5?
      end;
```

Anropet av hjälpfunktionen `findSpace(text,pos)` beräknar positionen för det första blanktecknet som finns i den del av `text` som börjar vid position `pos`. Om det inte finns något blanktecken där så beräknar `findSpace` positionen omedelbart efter slutet på `text` (dvs. längden av `text`). Exempel: `findSpace("a b c",1) = 1`, `findSpace("a b c",2) = 3`, `findSpace("a b c",4) = 5`.

Tips: Tänk på att positioner i strängar räknas från 0 och att `String.substring(t,p,a)` ger en sträng med `a` tecken från strängen `t` med början vid position `p`.

Frågor för betyg 3

Om du ger rätt svar på 7 av de 10 frågorna i detta avsnitt så blir du godkänd med minst betyg **3**, annars blir du underkänd (**U**). Du kan inte kompensera ett dåligt resultat i detta avsnitt med poäng från frågorna för betyg **4** eller **5**.

1. Vilken typ skall stå vid `?1?` ovan?
   (A) `string list` (B) `string*int->list` (C) `strint*int->string`
   (D) `string*int->string list` (E) `'a list*int->'a list`
   Justification:

2. Vilket av följande måste ingå i förvillkoret vid `?2?` ovan? (Det kan alltså behövas ytterligare villkor.)
   (A) `pos = 0`
   (B) `size text > 0`
   (C) `pos < size text`
   (D) `pos <= size text`
   (E) ***Inget*** av villkoren (A)-(D) skall ingå i förvillkoret!
   Justification: Det fullständiga förvillkoret är att `pos` måste vara en position i strängen eller – för basfallet – omedelbart efter strängen. Dvs. `0 <= pos <= size text`. Att `pos` är 0 i anropet ifrån `listOfWords` gör inte att förvillkoret skall vara `pos = 0`, eftersom `pos > 0` i de rekursiva anropen!

3. Vilken kod skall stå vid `?3?` ovan?
   (A) `firstspace = 0` (B) `firstspace < pos` (C) `firstspace = pos`
   (D) `firstspace > pos` (E) ***Inget*** av (A)-(D) utan något annat.
   Justification: Om `firstspace = pos` så är tecknet vid aktuell position ett blanktecken och det skall man hoppa över.

4. Vilken kod skall stå vid `?4?` ovan?
   (A) `String.substring(text,pos,firstspace)`
   (B) `String.substring(text,pos,firstspace-pos)`
   (C) `String.substring(text,firstspace,pos-firstspace)`
   (D) `String.substring(text,0,firstspace)`
   (E) ***Inget*** av (A)-(D) utan något annat.
   Justification: (B) tar ut tecknen från aktuell position i strängen fram till nästa blanktecken. Aktuell position finns i `pos`, antalet tecken fram till nästa blanktecken är skillnaden mellan aktuell position och positionen för blanktecknet.

5. Vilken kod skall stå vid `?5?` ovan?
   (A) `listOfWords'(text,pos)`
   (B) `listOfWords'(text,firstspace-pos)`
   (C) `listOfWords'(text,pos+firstspace)`
   (D)
   ```
   listOfWords'(String.substring(text,firstspace,size text-firstspace),
                firstspace)
   ```
   (E) ***Inget*** av (A)-(D) utan något annat.
   Justification: Rätt svar är `listOfWords'(text,firstspace)`

6. Vilken variant skall stå vid `?6?` ovan?
   (A) `pos` (B) `pos - size text` (C) `size text - pos`
   (D) `size text` (E) ***Inget*** av (A)-(D) utan något annat.
   Justification: `size text` ändras inte och `pos` ökar i det rekursiva anropet.

7. I något sammanhang arbetar man med körjournaler för bilar som innehåller information om de resor som gjorts med en bil. Närmare bestämt skall det för varje resa finnas information om:

   - Datum
   - Syfte
   - Bilens mätarställning före och efter
   - Varifrån och vart man kört

   Dessutom skall det gå att läsa ut den totala körsträckan utan att behöva summera körsträckorna för varje enskild resa.

   Man vill definiera en ny datatyp som skall representera sådana körjournaler:

   ```
   datatype journal = Journal of ...
   ```

   Vilket av följande alternativ är ***mest rimlig*** som fortsättning av deklarationen?
   (A) `(string*string*int*int*string*string*int) list`
   (B) `string list*string list*int list*int list*string list*string list*int`
   (C) `string*string*int*int*string*string list*int`
   (D) `(string*string*(int*int)*(string*string)) list*int`
   (E) `(string*string*int*int*string*string) list`
   Justification: Information om varje resa skall hänga ihop. Det finns bara en total körsträcka.

8. Vad är värdet av uttrycket `foo([1,2,3],2)` om `foo` är definierad så här?

   ```
   fun foo(first::rest,1) = first
     | foo(first::rest,n) = foo(rest,n-1);
   ```

   (A) `[2,3]` (B) 0 (C) 1 (D) 2 (E) 3
   Justification:

9. What is a tight asymptotic bound on the runtime of the function `g` below? Assume that `g` is defined only for positive integer powers of 4.

   ```
   fun g(4) = 3
     | g(n) = let val v = 5 * g(n/4) in v * v end
   ```

   (A) $\Theta(\lg n)$ (B) $\Theta(n)$ (C) $\Theta(n^2)$ (D) $\Theta(n^{\log_4 5})$ (E) $\Theta(n^{\log_4 5} \cdot \lg n)$
   Justification: By Theorem 2 (case 2), as $f(n) = \Theta(1{+}1) = n^{\log_b a}$, for $a = 1 \wedge b = 4$.

10. What is a tight asymptotic bound on the runtime of the function `h` below? Let n be the number of elements of its list argument. Assume that the infix function `#` ***always*** takes time linear in the number of elements of its ***left*** argument.

    ```
    fun h([ ]) = [ ]
      | h([x]) = [x]
      | h(x1::x2::xs) = [x1,x2] # h(xs)
    ```

    (A) $\Theta(n)$ (B) $\Theta(n \cdot \lg n)$ (C) $\Theta(n^2)$ (D) $\Theta(n^3)$ (E) Another bound
    Justification: By induction.

Frågor för betyg 4

Om du fått minst betyg **3** genom dina svar på de föregående frågorna och dessutom svarar rätt på minst 3 av de 5 frågorna i detta avsnitt så blir du godkänd med minst betyget **4**. Du kan inte kompensera ett dåligt resultat i detta avsnitt med poäng från frågorna för betyg **3** eller **5**.

11. Datastrukturdefinitionen i fråga 7 behöver en datastrukturinvariant. Vilken del (vilka delar) av datastrukturen behöver ***inte*** (rimligen) nämnas i datastrukturinvarianten? (Dvs förekomster av denna del kan ha vilket värde som helst – även oberoende av resten av datastrukturen.)
    (A) Datum (B) Syfte (C) Mätarställningar (D) Total körsträcka
    (E) Flera av dessa behöver inte nämnas i datastrukturinvarianten.
    Justification: Fältet för datum måste ha ett innehåll som verkligen beskriver ett datum. Varje mätarställning före måste vara mindre än motsvarande mätarställning efter. Mätarställningarna för olika körningar får inte överlappa. Totala körsträckan måste vara summan av skillnaderna mellan respektive före- och eftermätarställningar. Däremot finns inget som begränsar vad syftet kan vara.

12. Här är några olika testfall för `listOfWords'`:

    (a) `listOfWords'("a",0) = ["a"]`

    (b) `listOfWords'("a b c",0) = ["a","b","c"]`

    (c) `listOfWords'("a b c",2) = ["b","c"]`

    (d) `listOfWords'("a  b    c",0) = ["a","b","c"]`

    Vilken av följande uppsättningar av testfall ger full kodtäckning i `listOfWords'` utan att innehålla något/några onödiga testfall? ("Onödig" betyder alltså att man kan utelämna testfallet och ändå få full kodtäckning.)

    (A) a (B) a, b (C) b (D) b, c, d (E) a, b, c, d
    Justification: (a) testar inte fallet när det finns ett blanktecken i strängen, (b), (c) och (d) var för sig ger full kodtäckning.

13. Titta på funktionen `bar`:

    ```
    fun bar(x,y) = if x<y then 0 else 1+bar(x-y,y);
    ```

    Här är fyra förslag till eftervillkor för `bar`. Alla är i någon mening korrekta (om vi antar att det finns ett lämpligt förvillkor), men vilka är ***acceptabla*** eftervillkor i en funktionsspecifikation enligt de principer vi gått igenom i kursen?

    (a) Man räknar hur många gånger man kan subtrahera `y` från `x` utan att det blir mindre än 0.

    (b) Resultatet av att dividera `x` med `y`.

    (c) `x` div `y`

(d) 0 om `x` är mindre än `y`. Annars 1+ rekursion med argumenten `x-y` och `y`.

(A) a och c (B) b och c (C) b och d (D) a, b och c (E) Alla fyra.
Justification: b och c svarar på frågan "vad?" medan a och d svarar på frågan "hur?"

14. What is a tight asymptotic bound on the runtime of the function `t` below? Assume that `t`(m, n) is only defined when m is a prime number and n is a non-negative integer.

```
fun t(m,0) = 2
  | t(m,n) = t(3,n-1) + t(3,n-1)
```

(A) $\Theta(n)$ (B) $\Theta(n \cdot \lg n)$ (C) $\Theta(m \cdot n)$ (D) $\Theta(n^2)$ (E) $\underline{\Theta(2^n)}$
Justification: By Theorem 1, for $a = 2 \wedge b = \Theta(1)$. Red herring: Argument `m` does not affect the runtime.

15. What is a possible tight asymptotic bound on the runtime of the help function `d` of the function `v` below, so that `v`(L) always takes time quadratic in the number $|L|$ of elements of L? Let m be the number of elements of the list argument of the function `d`. Assume that `d`(L) returns three sub-lists of list L, each of about $|L|/2$ elements.

```
fun v([]) = 17
  | v(x::xs) = let val (P,Q,R) = d(xs) in v(P) + v(Q) + x + v(R) end
```

(A) $\Theta(m)$ (B) $\Theta(m \cdot \lg m)$ (C) $\Theta(m^{\log_2 3})$ (D) $\Theta(m^{\log_2 3} \cdot \lg m)$ (E) $\underline{\Theta(m^2)}$
Justification: By Theorem 2 (case 3), as $f(n)$ then is $\Theta((n-1)^2 + 1)$, which dominates $n^{\log_b a} \approx n^{1.6}$, for $a = 3 \wedge b = 2$, where n is the number of elements of the argument of `v`.

Frågor för betyg 5

Om du fått minst betyg **4** genom dina svar på de föregående frågorna och dessutom svarar rätt på minst 3 av de 5 frågorna i detta avsnitt så blir du godkänd med betyg **5**. Du kan inte kompensera ett dåligt resultat i detta avsnitt med poäng från frågorna för betyg **3** eller **4**.

16. Titta på funktionen `flatten`:

```
fun flatten [] = []
  | flatten [x] = x
  | flatten ([]::rest) = flatten rest
  | flatten (first::rest) = (hd first):: flatten((tl first)::rest);
```

`flatten` är en – inte speciellt bra skriven – funktion som "plattar ut" en lista av listor, dvs sätter ihop elementen till en lång lista.
Exempel: `flatten [[1,2],[],[3],[4,5,6]] = [1, 2, 3, 4, 5, 6]`

En viktig fördel med listor som implementeras med cons-celler är att två olika listor kan dela samma cons-celler vilket sparar utrymme i datorn. En nackdel är att programmet under sin körning kan skapa cons-celler "i onödan" – dvs cons-celler som inte blir en del av resultatet.

Antag att man gör anropet `flatten [[1,2],[3,4]]`, vilket ger resultatet `[1,2,3,4]`. Frågan är nu: hur många cons-celler delar resultatet med argumentet till `flatten` och hur många cons-celler skapar `flatten` som ***inte*** ingår i resultatet?

(A) 0 resp. 0 (B) 0 resp. 4 (C) 2 resp. 0 (D) 2 resp. 1 (E) 2 resp. 2
Justification: Cons-cellerna i argumentets sista element (`[3,4]`) delas med resultatet. Cons-cellerna i det rekursiva anropet kommer inte med i resultatet.

17. Man matar in följande deklarationer till ett ML-system (i denna ordning):

```
val x = 1;
val y = x+1;
fun bar x = x+y;
val x = x+2;
```

Därefter ger man – i tur och ordning – ML-systemet uttrycken `x` och `bar 0` att beräkna. Vilka värden beräknas?
(A) 2 och 2 (B) 3 och 1 (C) 3 och 2 (D) 3 och 3 (E) 5 och 2
Justification:

18. Vilket av följande påståenden om funktioner (i ML) är ***felaktigt***?
(A) Funktioner behöver inte nödvändigtvis deklareras med `fun`.
(B) Program kan använda funktioner som data.
(C) Varje funktion har ett namn som man använder för att referera till den.
(D) Olika anrop av samma funktion kan ha argument med olika typ.
(E) Funktionsargument beräknas alltid innan funktionen anropas.
Justification: Anonyma funktioner, t.ex., har inte namn.

19. Vilket av följande påståenden om algoritmer är ***felaktigt***?
(A) I dagligt tal menar man med "algoritm" ofta en grundläggande byggsten eller mönster som används i programmering.
(B) Alla program implementerar algoritmer.
(C) Problemet som algoritmen arbetar med måste kunna representeras som data (tal).
(D) En algoritm avslutar alltid sitt arbete inom begränsad tid.
(E) Vid varje steg av algoritmen måste det vara entydigt bestämt vad som skall göras.

Justification: En algoritm måste ge ett resultat inom (ändlig) tid, men ett program behöver inte göra det– det kan t.ex. ha oändlig rekursion.

20. What is a tight asymptotic bound on the runtime of the function `w` below? Let n be the number of elements of its list argument. Assume that `d`(L) returns two sub-lists of list L, each of about half the length of L, ***always*** in constant time (under some miracle). In order to determine the runtime of the two calls to `@`, note that you must ***first*** determine the length of `w(A)`.

```
fun w([]) = []
  | w(x::xs) = let val (A,B) = d(xs) in (w(A) @ [x]) @ w(B) end
```

(A) $\Theta(n)$ (B) $\Theta(n \cdot \lg n)$ (C) $\Theta(n^2)$ (D) $\Theta(n^2 \cdot \lg n)$ (E) $\Theta(2^n)$
Justification: By Theorem 2 (case 2), as $f(n) = \Theta(1+n) = n^{\log_b a}$, for $a = 2 = b$, since $|\mathtt{w}(L)| = |L|$, by induction.