Tentamen (del 2) (4 högskolepoäng) i Programkonstruktion och datastrukturer (1DL201)

Pierre Flener

Onsdag 14 mars 2012, kl. 14:00 – 17:00, i Bergsbrunnagatan 15, sal 1

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

Hjälp: En av huvudlärarna kommer normalt att besöka skrivsalen c:a klockan 15:00.

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):

. . . . . .

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. Här följer ett fragment av en abstrakt datatyp för tabeller (implementerade som listor som vi talat om i kursen) och några funktioner som använder tabellerna. Programraderna är numrerade.

```
 1  abstype 'a table = Table of (string*'a) list
 2  with
 3    val empty = ...;
 4    fun toList(Table T) = T;

 5    fun insert(Table [], key, value) = Table [(key,value)]
 6      | insert(Table ((key',value')::rest), key, value) =
 7          if key = key' then
 8            Table((key,value)::rest)
 9          else
10            Table((key',value')::toList(insert(Table rest,key,value)));

11    fun exists(...) = ...;
12    fun search (...) = ...;
13    fun delete (...) = ...
14  end;

15  fun countup(T,k) =
16    if exists(T,k) then
17      insert(T,k,search(T,k)+1)
18    else
19      Table ((k,1)::toList T);

20  fun countdown(T,k) =
21    case search(T,k) of
22      1 => delete(T,k)
23    | n => insert(T,k,n-1);
```

   En av raderna i programmet är felaktig (i den meningen att programmet inte går att köra) genom att använda datastrukturen på ett felaktigt sätt. Vilken?
   (Tips: Programlogiken som sådan är korrekt. Du behöver alltså inte förstå vad programmet gör eller hur det fungerar.)
   (A) 4 (B) 10 (C) 17 (D) 19 (E) 23
   Justification: Konstruktorn `Table` får inte användas utanför den abstrakta datatypen.

2. Titta på denna funktionsdefinition:

```
fun klingsor(kundry,[amfortas])=amfortas
  | klingsor(kundry,titurel::gurnemanz)=
      kundry(titurel,klingsor(kundry,gurnemanz))
```

Vad är värdet av uttrycket `klingsor(op -,[1,2,3])`?
(A) 2 (B) 0 (C) ~ 2 (D) ~ 4 (E) ~ 6
Justification: Anropen av `klingsor` ger uttrycket `op -(1, op -(2, 3))`, som beräknas till $1 - (2 - 3) = 2$.

3. Man matar i tur och ordning in följande deklarationer & uttryck till ett ML-system:

```
val x = ref 7;
fun bump(x) = (x := !x-1);
x := 11;
bump(x);
!x;
```

Vad blir värdet av det sista uttrycket `!x`?
(A) `5` (B) `6` (C) `7` (D) `10` (E) `11`
Justification: Definitionen av `bump` påverkar inte `x`. `x` sätts till `11`, sedan anropas `bump` som sätter `x` till `10`.

4. 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 ***right*** argument. Assume that the input and output of function `h` always have the same length (this assumption was unfortunately missing in the version handed out at the exam, but it was announced during the exam; for those who answered this question wrong, the threshold on this part of the exam was lowered from 10 to 9).

```
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.

5. After performing the insertions of the keys 9, 5, 1, 3, 0, 4, in that order, into the initially empty red-black tree using the algorithm seen in the course, what is the number of ***black*** nodes in the resulting red-black tree?
(A) ≤ 1 (B) 2 (C) 3 (D) 4 (E) ≥ 5
Justification: The resulting red-black tree is `RBT (Black, RBT (Red, RBT (Black, RBT (Red, Void, 0, Void), 1, Void), 3, RBT (Black, Void, 4, Void)), 5, RBT (Black, Void, 9, Void))`.

6. After performing the insertions of the keys 9, 5, 1, 3, 0, 4, in that order, into the initially empty binomial ***max***-heap using the algorithm seen in the course, what is the ***rank*** of the binomial tree ***containing*** the node with key 4 in the resulting binomial heap?
(A) 0 (B) 1 (C) 2 (D) 3 (E) ≥ 4
Justification: The resulting heap is $[4(0), 9(3(1), 5)]$.

7. Which of the following statements on hashing under ***chaining*** are ***false***?

   (a) The maximum chain length grows ***on average*** in proportion to the load factor.

   (b) The average chain length grows ***on average*** in proportion to the load factor.

   (c) A hash table of m cells can store strictly more than m elements.

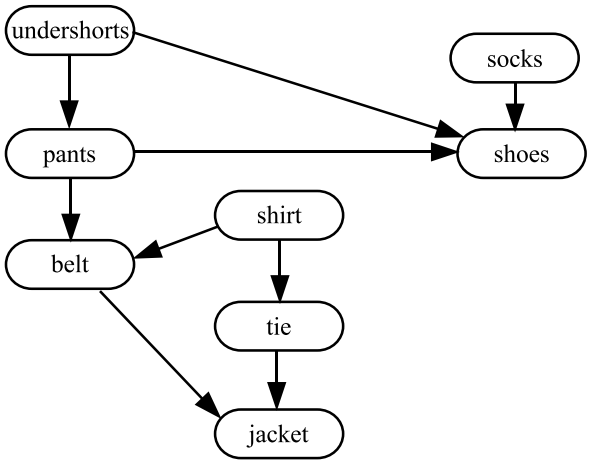
   (A) none (B) (a) only (C) (a) & (b) only (D) (b) only (E) all
   Justification: In the *average* case, the maximum chain length grows *slower* than the load factor.

8. Consider the hash table below of $m = 7$ cells, where $\perp$ denotes the element at a cell that was never used and Δ denotes a deleted element:

0	1	2	3	4	5	6
$\perp$	11	Δ	$\perp$	$\perp$	25	$\perp$

   Consider the ordinary hash function $hash'(key)$ = "the ***rightmost*** digit of *key*" under ***open addressing*** with the ***linear*** probing function $f(i) = i$. Perform ***no rehashing***. Assume that duplicate keys are ***not*** allowed. What is the number of probes made (that is, the number of values of i tried) when inserting 42?
   (A) ≤ 1 (B) 2 (C) 3 (D) 4 (E) ≥ 5
   Justification: Key 42 goes into cell 2 upon 2 probes, at indices $2+0 = 2$, $2+1 = 3$ for $i = 0, 1$, because under the given assumption we *must* probe beyond the Δ in order to make sure no 42 exists yet in the table. Red herring: The no-rehashing directive is irrelevant since there *is* space for another element.

9. Alfons needs to be careful when getting dressed in the mornings. Each edge from node u to node v in the acyclic directed graph below means that clothing item u must be put on before clothing item v:



Assume the graph is represented as an array of adjacency lists. Assume the array is indexed by ***decreasing*** alphabetic order on the node names. Assume each adjacency list is sorted by ***decreasing*** alphabetic order on the node names. What is the topological sorting order of the nodes, using the algorithm based on ***depth***-first search (DFS) seen in the course?

(A) shirt, socks, tie, undershorts, pants, belt, jacket, shoes
(B) shirt, socks, tie, undershorts, pants, shoes, belt, jacket
(C) socks, undershorts, pants, shoes, shirt, belt, tie, jacket
(D) undershorts, socks, shirt, tie, pants, shoes, belt, jacket
(E) Another order

Justification: By the DFS-based topological sort algorithm.

10. Reconsider the graph and its representation assumptions of Question 9. What is the ***breadth***-first search (BFS) order of this graph when starting from the undershorts and not making any restarts, using the algorithm seen in the course?

(A) undershorts, pants, belt, jacket, shoes
(B) undershorts, pants, belt, shoes, jacket
(C) undershorts, pants, shoes, belt, jacket
(D) undershorts, pants, shoes, belt, shoes, jacket
(E) undershorts, shoes, pants, belt, jacket

Justification: By the BFS algorithm.

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. Vilket av dessa påståenden stämmer – i princip – ***inte*** in på abstrakta datatyper?
(A) Dataabstraktion innebär att man döljer datarepresentationen.
(B) För att förstå hur man använder en abstrakt datatyp räcker det med att förstå hur dess gränssnitt fungerar.
(C) Om man ändrar datarepresentationen i en abstrakt datatyp så måste man i allmänhet också göra ändringar i övriga delar av programmet.
(D) En abstrakt datatyp förhindrar direkt åtkomst till en datastruktur.
(E) Abstrakta datatyper är i första hand ett sätt att organisera program och man skulle kunna programmera på samma sätt även om deklarationen `abstype` inte fanns i ML.
Justification: Själva tanken med abstrakta datatyper är att hur de används (gränsytan) skall vara oberoende av hur de är implementerade (representationen). Ändringar av representationen skall alltså inte påverka gränsytan och därmed så behöver inte programmet utanför den abstrakta datatypen ändras.

12. Funktionen `find` nedan (med numrerade rader) är tänkt att beräkna numret på den första rad i en fil som är lika med en viss sträng. `find` har följande specifikation:
`find(filename,s)`
TYPE: `string*string->int`
PRE: `filename` är namet på en fil som går att läsa.
POST: Numret på den första rad i `filename` som är lika med `s`, och 0 om ingen sådan rad finns.

```
 1 fun find(f,s) =
 2   let
 3     fun find'(f',s,n) =
 4       if TextIO.endOfStream f' then
 5         (TextIO.closeIn f'; 0)
 6       else
 7         if valOf(TextIO.inputLine f') = s^"\n" then
 8           n
 9         else
10           find'(f',s,n)
11   in
12     find'(TextIO.openIn f, s,1)
13   end
```

Det finns felaktigheter eller något som saknas på en eller flera rader. Vilken/vilka?
(A) 5 & 8 (B) 5 & 10 (C) 8 & 10 (D) 8 & 10 & 12 (E) 12
Justification: Filen behöver stängas även på rad 8. Radräknaren ökas inte på rad 10.

13. Which of the following statements on walks of a right-rotatable ***balanced*** binary ***search*** tree T of at least two elements, any two of them being ***distinct***, are ***true***?

    (a) The inorder walk of T takes time exponential in the height of T.

    (b) The inorder walk of T lists the elements of T by strictly increasing keys.

    (c) The inorder walk of T is unchanged after a right rotation at the root of T.

    (A) all (B) (a) & (b) (C) (b) only (D) (b) & (c) (E) (c) only
    Justification: Because the height of a balanced tree is logarithmic in its number n of elements and the inorder walk takes time *linear* in n; because of the search invariant; and because it is indeed the *in*order walk that is preserved by rotations.

14. What is the total number of multiplications made when evaluating $\texttt{p}(n)$ using the definition of `p` below? Assume that a pre-condition for $\texttt{p}(n)$ is that n is a positive integer power of 2.

```
fun p(2) = 1
  | p(n) = let val d = 4 * p(n div 2) in d * d end
```

    (A) $\log_2 n$ (B) $4 \cdot \log_2 n - 1$ (C) $\frac{1}{6} \cdot n^2 - \frac{2}{3}$ (D) $\frac{1}{12} \cdot n^2 - \frac{1}{3}$ (E) Another answer
    Justification: It is $2 \cdot (\log_2 n - 1)$. By induction; it is the closure of the following divide-and-conquer recurrence for the number $M(n)$ of multiplications made when evaluating $\texttt{p}(n)$:

$$M(n) = \begin{cases} 0 & \text{if } n = 2 \\ 1 \cdot M(n/2) + 2 & \text{if } n = 2^k \text{ for } k > 1 \end{cases}$$

15. What is a tight asymptotic bound on the runtime of the function `t` below? Assume that a pre-condition for $\texttt{t}(m, n)$ is that n is a non-negative integer.

```
fun t(m,0) = 2
  | t(m,n) = t(m+3,n-1) + t(m+4,n-1)
```

    (A) $\Theta(n)$ (B) $\Theta(n \cdot \lg n)$ (C) $\Theta(m \cdot n)$ (D) $\Theta(n^2)$ (E) $\Theta(2^n)$
    Justification: By Theorem 1, for $a = 2 \wedge b = 0$. Red herring: The first argument does not affect the runtime.

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 igen på funktionen från uppgift 2:

```
fun klingsor(kundry,[amfortas])=amfortas
  | klingsor(kundry,titurel::gurnemanz)=
      kundry(titurel,klingsor(kundry,gurnemanz))
```

Vilken typ har `klingsor`?
(A) `('a * 'a -> 'a) * 'a list -> 'a`
(B) `('a * 'b -> 'b) * 'a list -> 'b`
(C) `('a * 'b -> 'c) * 'a list -> 'c`
(D) `(int * int -> int) * int list -> int`
(E) `int * int list -> int`
Justification: Av första klausulen till `klingsor` så kan man se att andra argumentet till `klingsor` måste vara en lista med element av samma typ som värdet av `klingsor`. Alltså är både alternativ (B) och (C) fel. Av sista raden i funktionsdefinitionen kan man se att `kundry` (första argumentet till `klingsor`) är en funktion. Alltså är alternativ (E) fel. Återstår (A) och (D). `klingsor` innehåller inga funktionsanrop som kan begränsa typen av data till `int`, alltså är det rätta svaret (A).

17. Vilket av följande påståenden om referenser i ML är ***felaktigt***?
(A) Varje referens måste tilldelas ett bestämt värde när den skapas.
(B) Att ändra en datastruktur genom att ändra en referens som finns i den kan medföra att andra datastrukturer också ändras.
(C) Användning av referenser är ett naturligt inslag i funktionell programmering.
(D) Man kan ha en referens som hänvisar till en annan referens.
(E) Även om `!x = !y` så kan `x` och `y` vara olika.
Justification: En huvudprincip i funktionell programmering är att funktioner endast beräknar värden och inte har sidoeffekter. Även om sidoeffekter är möjliga i vanliga funktionella språk så är det inte naturligt att utnyttja dem. Meningsfull användning av referenser kräver användning av sidoeffekter.

18. Consider the hash table below of $m = 7$ cells, where Δ denotes a deleted element:

0	1	2	3	4	5	6
Δ	11	22	33	Δ	55	Δ

Consider the ordinary hash function $hash'(key)$ = "the ***leftmost*** digit of key" under ***open addressing***. Perform ***no rehashing***. Assume that duplicate keys ***are*** allowed. What is the difference in the numbers of probes made (that is, the number of values of i tried) when ***trying to*** insert the key 14 under quadratic probing (with the probing function $f(i) = i^2$) compared to linear probing (with $f(i) = i$)?

(A) 0 (B) 1 (C) 2 (D) 3 (E) ≥ 4

Justification: Under linear probing, key 14 goes into cell 4 upon 4 probes (at indices $1, 2, 3, 4$ for $i = 0, 1, 2, 3$), because under the given assumption we *need not* probe beyond the first Δ in order to make sure no 14 exists yet in the table; under quadratic probing, the insertion fails upon 7 probes (at indices $1, 2, 5, 3, 3, 5, 2$ for $i = 0, 1, 2, 3, 4, 5, 6$).

19. Which of the following statements on trees are ***false***?

    (a) A non-empty binary tree where all nodes have 0 or 2 children has one more leaf node than non-leaf nodes.

    (b) Every binomial tree is also a search tree, under some suitable generalisation of binary search trees to k-ary search trees, where $k \geq 2$ is a constant.

    (c) Every binomial tree is also a balanced tree, under some suitable generalisation of the red-black balancing invariants to k-ary trees, where $k \geq 2$ is a constant.

    (A) None (B) (c) only (C) (b) and (c) (D) All (E) Another answer

    Justification:
    (a): True. Let $L(n)$ be the number of leaf nodes of such a tree with n non-leaf nodes: $L(n) = \begin{cases} 2 & \text{if } n = 1 \\ L(n-1)+1 & \text{if } n > 1 \end{cases} = n + 1$, to be proved by induction on n.
    (b): False, because the elements of a binomial tree do not need to satisfy any invariant (such as a heap invariant), hence no such generalisation can exist.
    (c): False, because a binomial tree of rank k has k subtrees of heights $k-1, k-2, \ldots, 0$ respectively, hence for large enough k there will always be subtrees whose heights differ too much.

20. What is a tight asymptotic bound on the runtime of the function `k` below? Let n be the number of elements of its list argument. Assume that `d`(L) returns four sub-lists of list L, each of about ***half*** the length $|L|$ of L, ***always*** in $\Theta(|L|)$ time. In order to determine the runtime of the three calls to `@`, note that you ***must first*** determine the length of `k(X)` in terms of the length of `X`.

```
fun k([]) = []
  | k(x::xs) = let val (P,Q,R,S) = d(xs) in k(P) @ k(Q) @ x::k(R) @ k(S) end
```

    (A) $\Theta(n)$ (B) $\Theta(n \cdot \lg n)$ (C) $\Theta(n^2)$ (D) $\Theta(n^2 \cdot \lg n)$ (E) Another bound

    Justification: By Theorem 2 (case 2), as $|\mathtt{k}(L)| = \Theta(|L|^2)$ (by case 1 for $a = 4 \wedge b = 2 \wedge f(|L|) = \Theta(1)$), so that $f(n) = \Theta\left(3 \cdot \left(\frac{n}{2}\right)^2 + n + 1\right) = \Theta(n^{\log_b a}) = \Theta(n^2)$, for $a = 4 \wedge b = 2$.