TENTAMEN: Algoritmer och datastrukturer

Läs detta!

- *Uppgifterna är inte avsiktligt ordnade efter svårighetsgrad.*
- Börja varje uppgift på ett nytt blad.
- Skriv ditt idnummer på varje blad (så att vi inte slarvar bort dem).
- **Skriv rent dina svar. Oläsliga svar r ä t t a s e j!**
- Programmen skall skrivas i Java 5 eller senare, vara indenterade och kommenterade, och i övrigt vara utformade enligt de principer som lärts ut i kursen.
- Onödigt komplicerade lösningar ger poängavdrag.
- Programkod som finns i tentamenstesen behöver ej upprepas.
- Givna deklarationer, parameterlistor, etc. får ej ändras, såvida inte annat sägs i uppgiften.
- Läs igenom tentamenstesen och förbered ev. frågor.

I en uppgift som består av flera delar får du använda dig av funktioner klasser etc. från tidigare deluppgifter, även om du inte löst dessa.

Lycka till!

Uppgift 1

Välj **ett** svarsalternativ på varje fråga. Motivering krävs ej. Varje korrekt svar ger två poäng. Garderingar ger noll poäng.

1. Vilka av metoderna 1-3 måste vara implementerade för typen T vid användning av resp. standardklass A-C i Java? Välj det alternativ som anger ett nödvändigt och tillräckligt urval av metoder.

A. `TreeSet<T>`	1. `equals`
B. `HashSet<T>`	2. `compareTo` eller `compare`
C. `ArrayList<T>`	3. `hashCode`

   a. A:1, B:2, C:2
   b. A:1+2, B:1+2, C:1
   c. A:3, B:2, C:1
   d. A:1+2, B:1+3, C:1
   e. A:2, B:3, C:1

2. Om fältdubblering används vid implementering av en hashtabell och kvadratisk sondering används, vilket genomsnittligt minnesutnyttjande får man ungefär i tabellen?
   a. 10%
   b. 25%
   c. 38%
   d. 50%
   e. 75%

3. Ange en minsta övre begränsning för f(n), definierad av ekvationerna

   $$f(0) = 0$$
   $$f(n) = 2f(n-1) + 1 \qquad (n > 0)$$

   a. $O(2^n)$
   b. $O(n \log n)$
   c. $O(n^2)$
   d. $O(\log n)$

4. Sorteringsalgoritmer som baseras på successiva byten av närliggande element har för genomsnittliga indata tidskomplexiteten
   a. $O(n \log n)$
   b. $\Omega(n^2)$
   c. $O(n^2)$

forts.

5. Metoden `getReverse` tar en enkellänkad lista som argument och returnerar innehållet i omvänd ordning i en standardlista. Vilken av de fyra metoderna är korrekt implementerad?

```
public class ListNode {
    int element;
    ListNode next;
}

public class Lists {
    private static ArrayList<Integer> result = new ArrayList<Integer>();

    public static List<Integer> getReverse1(ListNode l) {
        if ( l == null )
            return result;
        else {
            getReverse1(l.next);
            result.add(l.element);
            return result;
        }
    }

    public static List<Integer> getReverse2(ListNode l) {
        List<Integer> result = new ArrayList<Integer>();
        if ( l == null )
            return result;
        else {
            getReverse2(l.next);
            result.add(l.element);
            return result;
        }
    }

    public static List<Integer> getReverse3(ListNode l) {
        if ( l == null )
            return new ArrayList<Integer>();
        else {
            List<Integer> result = getReverse3(l.next);
            result.add(l.element);
            return result;
        }
    }

    public static List<Integer> getReverse4(ListNode l) {
        if ( l == null )
            return null;
        else {
            List<Integer> result = getReverse4(l.next);
            result.add(l.element);
            return result;
        }
    }
}
```

a. `getReverse1`
b. `getReverse2`
c. `getReverse3`
d. `getReverse4`

(10 p)

Uppgift 2

Följande typ kan användas för att representera noder i ett binärt sökträd.

```
public class TreeNode {
    int element;
    int size;
    TreeNode left;
    TreeNode right;
}
```

Förutom referenser för vänster och höger delträd, och dataelement, finns även en instansvariabel för att lagra trädets storlek. Storleken skall ange antalet noder i trädet som har sin rot i noden i fråga.

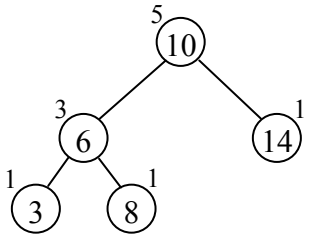
a) Definiera en konstruktor till `TreeNode` som fungerar så att variabeln `size` i alla noder i ett träd innehåller delträdets storlek (antalet noder), se exempel nedan.

```
public TreeNode(TreeNode left, TreeNode right,int x)
```

Ex. Uttrycket

```
new TreeNode(
    new TreeNode(new TreeNode(null,null,3),
                 new TreeNode(null,null,8),
                 6),
    new TreeNode(null,null,14),
    10);
```

skall ge trädet



(4 p)

b) Skriv en rekursiv funktion

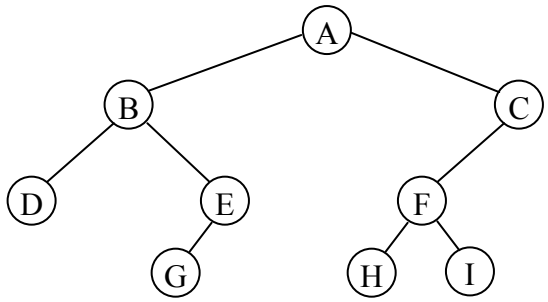
```
public static int findNth(int n,TreeNode t)
```

som returnerar det `n`:te minsta elementet i sökträdet `t`. Om `t` är tomt (`null`) skall undantaget `IllegalArgumentException` kastas.

(8 p)

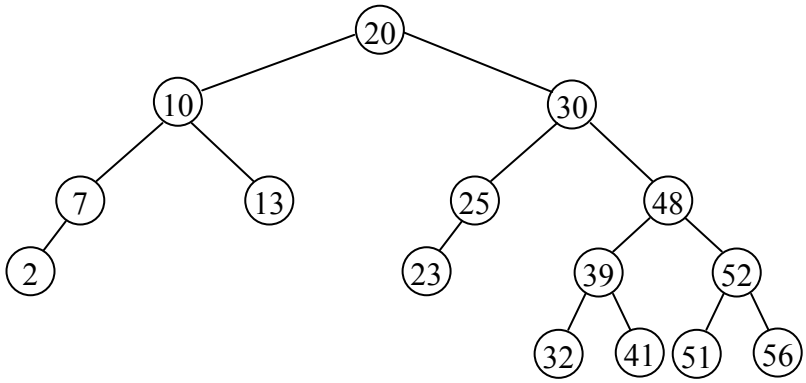
Uppgift 3

a) Ange i vilken ordning noderna besöks vid preorder, postorder, resp. inorder genomlöpning av trädet



(6 p)

b) Visa hur AVL-trädet nedan ser ut efter insättning av 53 och lämplig AVL-rotation



(4 p)

Uppgift 4

a) Visa med en figur hur en initialt tom hashtabell med 11 platser ser ut efter sekvensen

```
add(21); add(10); add( 43); remove(10); add(43);
```

Kvadratisk sondering används vid insättning och hashfunktionen definieras hash(x) = x mod 11. Förklara vad som händer vid varje anrop ovan.

(3 p)

b) Hur många platser kan ockuperas i tabellen i a innan omhashning till en större tabell måste ske? Vilken är den minsta storlek man då bör välja om vi antar att det nya fältet skall vara minst dubbelt så stort?

(2 p)

Uppgift 5

a) Vilka villkor måste vara uppfyllda för att ett träd skall vara en binär hög?

(1 p)

b) Visa i trädform hur den binära högen

	7	10	9	16	13	14	15	23	22	15	46	18	15	21	19
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*

ser ut efter varje anrop i sekvensen (du skall alltså rita tre träd)

```
insert(25); deleteMin(); insert(5);
```

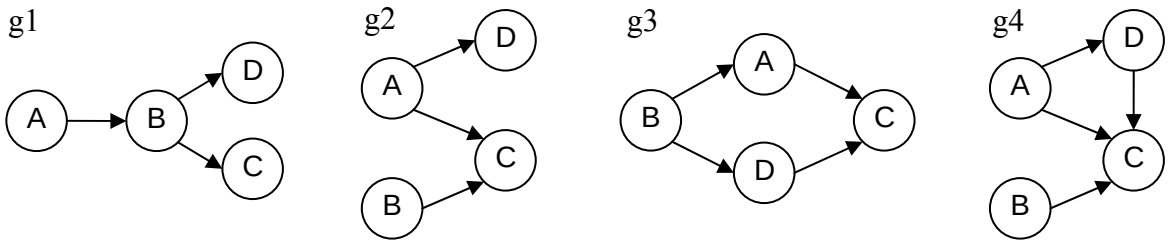
(3 p)

c) En lista innehåller N heltal. Antag att man vill kopiera elementen till ett fält så att fältet blir en binär hög. Ange ordokomplexiteten för worst case om elementen placeras i fältet med operationen `insert`. Finns det något effektivare sätt? Vilket i så fall, och vad blir WC då?

(2 p)

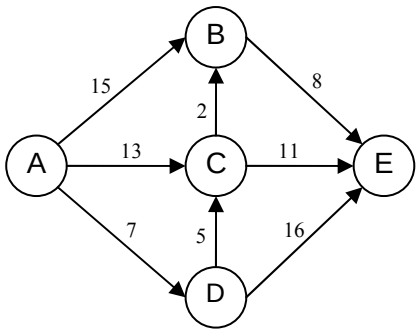
Uppgift 6

a) För en av graferna g1 – g4 är sekvenserna ABCD och BADC möjliga topologiska ordningar av noderna. Vilken är grafen? Motivera svaret! Lista övriga topologiska ordningar för grafen i fråga.



(3 p)

Betrakta grafen



b) I vilken ordning besöks noderna vid bestämning av de kortaste oviktade avstånden från A till samtliga övriga noder?

(1 p)

c) I vilken ordning besöks noderna i Dijkstras algoritm vid bestämning av de kortaste viktade avstånden från A till samtliga övriga noder?

(1 p)

d) Ange de kortaste viktade avstånden från A till samtliga övriga noder.

(1 p)

e) Vilken datastruktur, förutom graftabellen, är vital i Dijkstras algoritm?

(1 p)

Uppgift 7

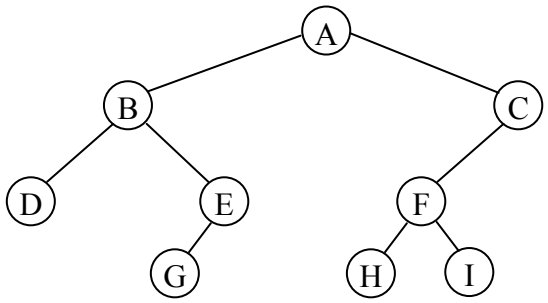
Antag att noder i ett binärt träd representeras med typen

```
public class TreeNode<T> {
    T element;
    TreeNode<T> left;
    TreeNode<T> right;
}
```

Implementera metoden

```
public static <T> List<T> getLevelOrder(TreeNode<T> t)
```

som givet ett träd returnerar en lista av elementen i trädets noder i nivåvis ordning.
Exempel: Om trädet är



så skall `getLevelOrder` ge en lista med elementen A,B,C,D,E,F,G,H,I (i nämnd ordning). För full poäng på uppgiften krävs att du använder en lämplig datastruktur.

(10 p)