# Tentamen i TDP004
# Objektorienterad Programmering
# Teoretisk del

| | |
|---|---|
| Datum: | 2009-08-24 |
| Tid: | 14-18 |
| Plats: | SU-salar i B-huset. |
| Jour: | Per-Magnus Olsson, tel 285607<br>Jourhavande kommer att besöka skrivsalarna ungefär varje timme under skrivtiden. |
| Hjälpmedel: | Teoretisk del: Inga.<br>Praktisk del: Den C++ information som finns i systemet. |
| Betygsättning: | Max antal poäng: 48 med 24 poäng vardera på teori och praktikdel. |

| Poäng | **Betyg** |
|---|---|
| 41-48 | **5** |
| 33-40 | **4** |
| 25-32 | **3** |
| 0-24 | **U** |

Anvisningar: Börja med den teoretiska delen. När du är klar med den lämnar du in den och får den praktiska delen. När du har lämnat in den teoretiska delen kan du inte återvända till den.

Skriv svaret på varje teoretisk uppgift på ett separat blad.

Uppgifterna är inte ordnade efter svårighetsgrad.

Lycka till!

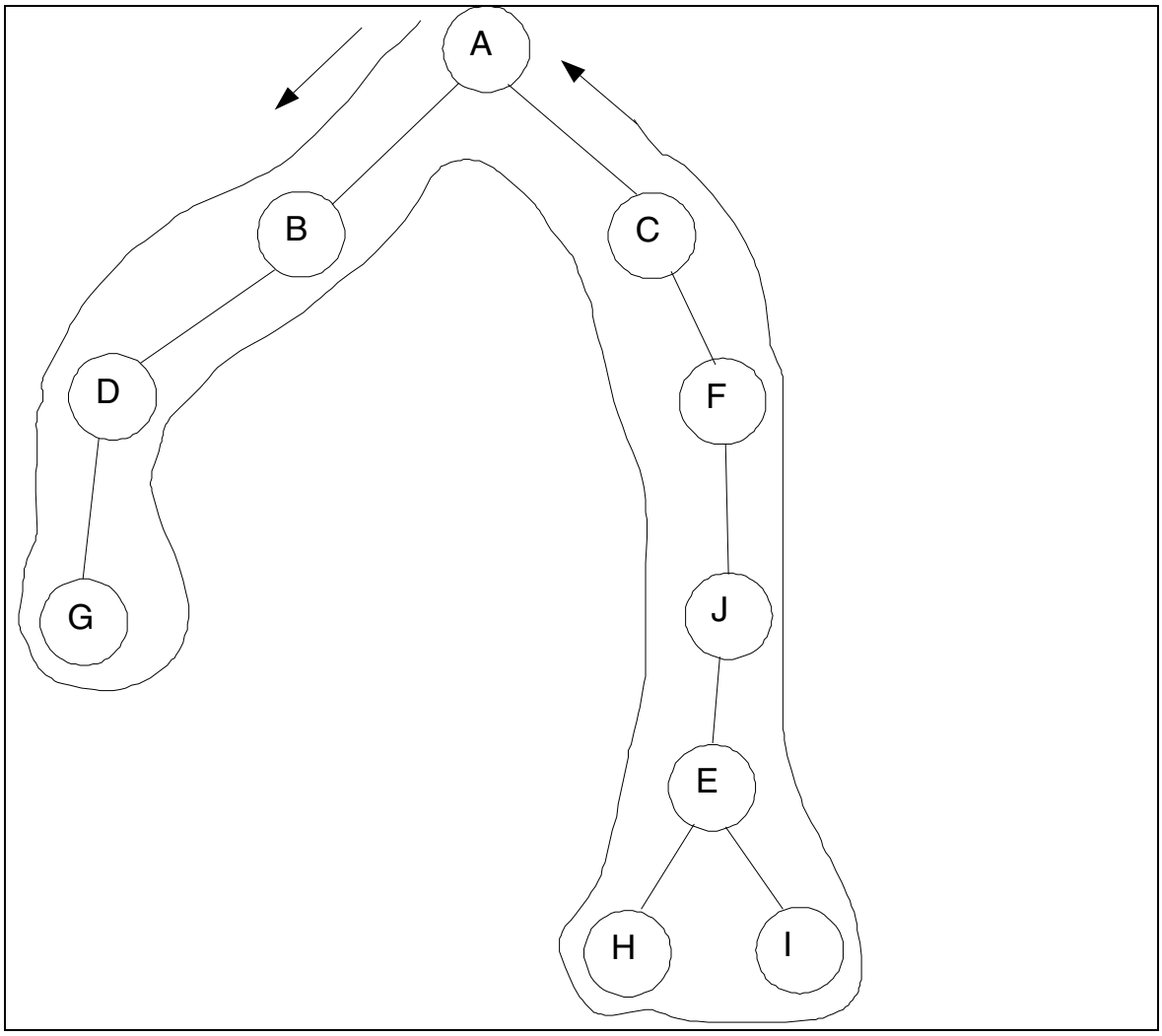
# TDP004 Objektorienterad Programmering
## Teoretisk del

1. Inom objektorientering används ofta termerna abstraktion, arv och inkapsling. Ge exempel på andra sammanhang där termerna används och motivera kortfattat dina exempel. (6p)

2. a) I större projekt är det vanligt att en kodstandard används. Vilka 4 saker tycker du är viktigast i en kodstandard? Du får ett poäng per sak samt en poäng per sak för bra motivering. (8p)

3. Ge förslag på sätt att rätta eventuella fel på följande kodrader. Är det inget fel så behöver du inte skriva något. (3p)
   a) `int int_p= new int;`
   b) `double & double_r;`
   c) `cout >>` ”Hejsan”;

4. (Svårare uppgift) Inom matematiken finns ett koncept som kallas *thread index*, vilket betyder att man lagrar i vilken antalet barn för alla noder i ett träd som symboliserar en graf. I bilden nedan markeras en nod med en rund ring och en nods barn är alla noder som ligger ”under” den i trädet. Man får reda på antalet barn genom att gå runt trädet i den riktning som pilarna i bilden visar: först A, sedan B, D, G, C osv tills man kommer tillbaka till A.
   **Uppgiften fortsätter på nästa sida.**

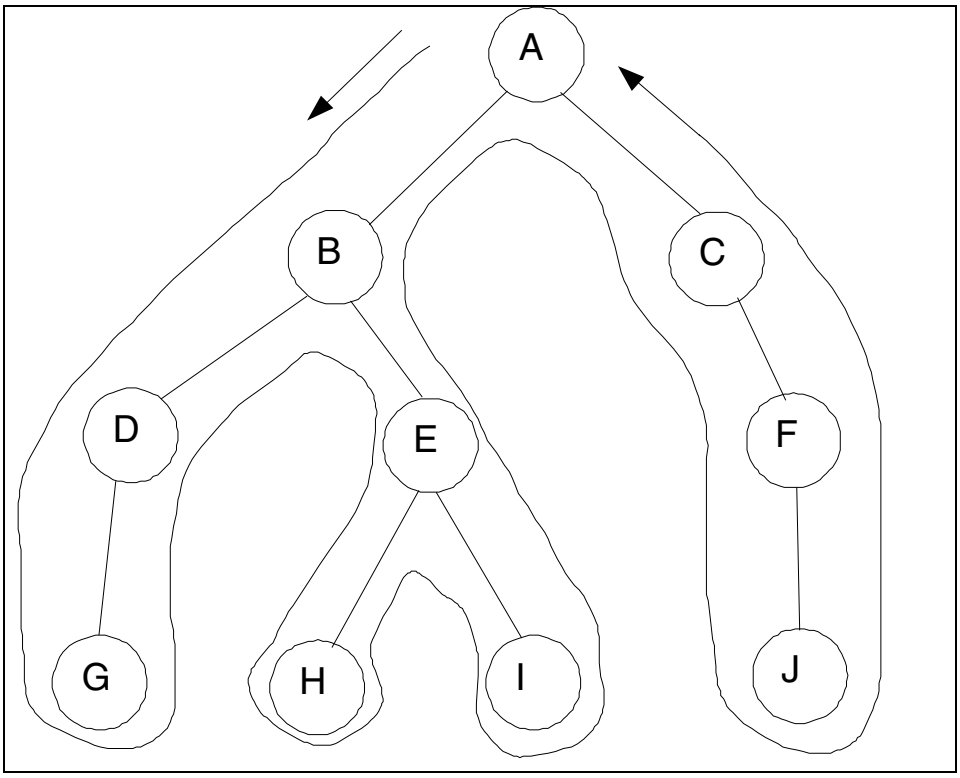



| A | B | D | G | C | F | J | E | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 2 | 1 | 0 | 5 | 4 | 3 | 2 | 0 | 0 |

Man tänker sig att man har en viss ordning på noderna, vilket är den ordning som man kommer till dem när går igenom trädet, se tabellen ovan. Dock vet man inte hur många barn en nod har förrän man kommer till noden för *sista* gången. Därför måste man vänta med att lägga antalet barn tills man besöker noden för sista gången. När man har gått igenom trädet och kommit tillbaka till nod A vet man antalet barn för alla noder, vilket visas i ovanstående tabell. Till exempel har nod J har 3 barn: noderna E, H och I, varav nod E i sin tur har 2 barn: noderna H och I.

a) Ge förslag på hur man kan förknippa noden med antalet barn. (2p)
b) Välj en lämplig datastruktur för att lagra thread index. Även om tabellen ovan använder en tabell-liknande struktur behöver du inte göra det. Du ska endast ta hänsyn till ovanstående information. Motivera ditt svar. (2p)

c) Efter att thread index har beräknats används det som indata till funktioner som ofta modifierar trädet så att antalet barn ändras. Då måste man naturligtvis uppdatera thread index. I exemplet nedan är delträdet bestående av noderna E, H and I är flyttat till att ligga under nod B. Det är alltid ett enda delträd som flyttas.




| A | B | D | G | E | H | I | C | F | J |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 5 | 1 | 0 | 2 | 0 | 0 | 2 | 1 | 0 |

Ändringen ger ovanstående thread index där noderna E, H och I har flyttats. Eftersom sådana uppdateringar måste gå väldigt snabbt är det viktigt att välja en passande datastruktur för att spara thread index. Trädet kan innehålla miljontals noder så man vill undvika att gå igenom hela trädet igen efter en uppdatering. Istället håller man reda på det delträd som flyttades och flyttar detta delträd till sin nya plats (och uppdaterar antalet barn för alla involverade noder. I bilden ovan har trädet uppdaterats och då måste antalet barn minskas för noderna C, F och J och antalet barn ökas för nod B). Nu ska du igen välja en lämplig datastruktur och den här gången ska du även ta hänsyn till informationen i deluppgifterna a) och b). Motivera ditt svar. (3p)