Tentamen - Datastrukturer, algoritmer och programkonstruktion.

DVA104

Akademin för innovation, design och teknik

Måndag 2015-06-01

Skrivtid: 14.30-19.30
Hjälpmedel: Bok relaterad till ämnet – ej digital (inga föreläsningsanteckningar eller egna anteckningar är tillåtna)
Lärare: Caroline Uppsäll, 021-101456

Betygsgränser

Betyg 3:	23p - varav minst 6 poäng är P-uppgifter
Betyg 4:	33p
Betyg 5:	40p
Max:	44p - varav 11 poäng är P-uppgifter

Allmänt

- Kod skriven i tentauppgifterna är skriven i pseudokod.
- På uppgifter där du ska skriva kod (P-uppgifter) ska koden skrivas i det språk som var aktuellt då du tog kursen (C eller Ada). Ange högst upp på bladet vilket språk koden är skriven i.
- Skriv endast en uppgift per blad
- Skriv bara på ena sidan av pappret.
- Referera inte mellan olika svar.
- Om du är osäker på vad som avses i någon fråga, skriv då vad du gör för antagande.
- Oläsliga/Oförståeliga svar rättas inte.
- Kommentera din kod!
- Eventuellt intjänade bonuspoäng kan endast användas på ordinarie tentamenstillfälle.
- Tips: Läs igenom hela tentan innan du börjar skriva för att veta hur du ska disponera din tid.

Lycka till!

Uppgift 0: *(0p)*

Läs noga igenom instruktionerna på förstasidan och följ dem!

Uppgift 1: *(7p)*

a) Beskriv kortfattat hur rekursion fungerar – glöm inte att få med de fyra viktiga parametrar som måste finnas för att en funktion ska vara rekursiv. *(2p)*
b) Vad innebär det att en sorteringsalgoritm är naturlig? *(1p)*
c) Vad innebär det att ett binärt träd är balanserat? *(1p)*
d) Nämn två olika sätt att implementera en LIFO-kö på. *(1p)*
e) Nämn minst två anledningar till varför man i en ADT ska skilja på interface, implementation och användning. *(1p)*
f) Vad innebär det att en algoritm har linjär komplexitet? *(1p)*

Uppgift 2: *(3p)*

Vi har i kursen diskuterat arraybaserade cirkulära köer. Antag från en början tom sådan typ av kö med 7 platser. Funktionen för att lägga till data i kön heter `enqueue`, där anges det nya datat som parameter. Funktionen för att ta bort data ur kön heter `dequeue`. Till kön finns det två iteratorer (***`front`*** och ***`back`***).

Illustrera steg för steg hur den arraybaserade cirkulära kön ser ut när nedanstående anrop körs, markera tydligt vart front och back befinner sig.
Uppstår det några problem?

```
enqueue(1); enqueue(2); enqueue(3); dequeue(); enqueue(4);
dequeue(); enqueue(5); dequeue(); dequeue(); enqueue(6);
enqueue(7); enqueue(8); enqueue(9); enqueue(10); enqueue(11);
dequeue()
```

Uppgift 3: *(3p)*

Ange för nedanstående delar av olika algoritmer en uppskattning av exekveringstiden i form av Big-Oh/Ordo notation. Motivera också din uppskattning.
a)

```
for(int i=0; i<n; i++)
          x=x+1;
 for(int j=1; j<n/2; j++)
          x=x+2;
```

b)

```
for(int i=n; i>=1; i=i/2)
{
          for(int j=1; j<n; j++)
                   x = x+1;
}
```

c)

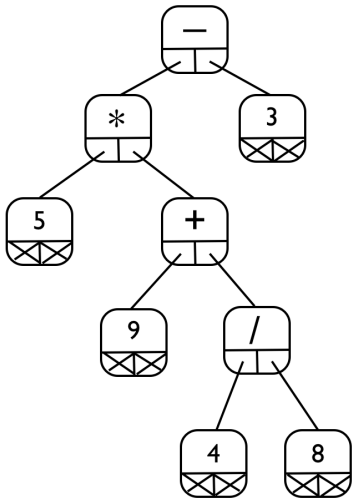
```
for(int i=1; i<=42; i=i*2)
{
        for(int j=0; j<i; j++)
                x = x+1;
}
```

Uppgift 4: *(4p)*

Vi kan använda binära träd för att representera algebraiska uttryck. Dessa träd kallas ”expressions tree”. Om man använder metoden `printPostOrder` för att skriva ut trädet får vi ett aritmetiskt utryckt i postfix notation.

Se implementation av funktionen nedan samt en grafisk representation av ett träd.

```
printPostOrder(Node subTree ) //delträd skickas in
        if subTree != NULL then //inte lika med NULL
                printPostOrder(subTree.left)
                printPostOrder(subTree.right)
                print subTree //skriv ut aktuell nod
        end if
```



a) Vilken blir uttrycket för trädet ovan då det skrivs ut med funktionen `printPostOrder`. *(1p)*

b) För att beräkna aritmetiska uttryck i postfix notation används datastukturen stack. Visa hur stacken förändras när du beräknar postfix uttrycket som du fått fram i uppgift a. Lyckas du inte lösa uppgift a väljer du istället själv ett artimetiskt uttryck i postfix och visar hur beräkningen med hjälp av stacken sker. *(3p)*

Exempel:

Vanligt räkneuttryck	Postfix
a + b	ab+
a + b * c	abc*+
(a + b) * c	ab+c*
(a – (b + c)) * (c – d)	abc+-cd-*

Uppgift 5: *(5p)*

a) Skriv koden för en optimerad bubblesort. Algoritmen ska vara optimerad på två olika sätt jämfört med standardimplementationen. Du ska även beskriva vilka de två optimeringarna är. ---***P-uppgift*** *(4p)*

b) Hur påverkar optimeringarna komplexiteten på algoritmen? *(1p)*

Uppgift 6: *(6p)*

a) Beskriv sorteringsalgoritmerna mergesort och quicksort med ord. *(4p)*

b) Påverkas tidskomplexiteten för någon/några av algoritmerna i a om elementen redan är sorterade. Oavsett om du kommer fram till att svaret är Ja eller Nej ska du motivera ditt svar. *(2p)*

Uppgift 7: *(6p)*

a) Antag följande träd och nod:

```
BTree
    Node root

Node
    Integer data
    Node leftChild
    Node rightChild
```

Skriv den rekursiva funktionen som tar reda på hur djupt ett träd är. Du kan anta att trädet är sorterat och att alla länkar som inte pekar på någon nod är satta till NULL. Funktionen ska returnera det aktuella djupet på trädet (alltså djupet på det djupaste delträdet). Om trädets root ligger på nivå 1. ---***P-uppgift*** *(4p)*

b) Beskriv två olika sätt att balansera ett binärt sökträd på. Det räcker inte att endast ange namnet på balanseringen utan en beskrivning av hur balanseringen går till måste ges. *(2p)*

Uppgift 8: *(5p)*

a) Förklara vad som är olämpligt i följande exempel. Beskriv också hur man istället bör göra. Motivera. *(2p)*

   Exempel: En idrottsförening vill ha ordning på alla sina medlemmar. Idrottsföreningen väljer att lagra medlemmarna (som är av typen Person-objekt) i en hashtabell. Som hashnyckel väljer man de två första siffrorna i medlemmens personnummer.

b) Antag att man har en hashtabell där elementeten lagras i en array av storleken 11. Hash-funktionen är h(x) = x % 11. Öppen adressering (linear probing) används för att hantera kollisioner. Antag att man satt in följande element (i angiven ordning) i hashtabellen: 34, 45, 3, 67, 65, 19, 1, 17.

   Hur ser tabellen ut efter dessa insättningar? *(2p)*

c) Om man önskar ta bort elementet 34 och bara lämnar dess plats i tabellen tom uppstår det problem. Vilket är problemet? *(1p)*

Uppgift 9: *(5p)*

a) Antag att en vän inte gått Algoritmer och Datastruktur kursen och anropar binärsökningsfunktionen men en osorterad array. Vad kommer hända? Kraschar programmet? Hur påverkar detta sökningen? Motivera med exempel. *(2p)*

b) Skriv koden för funktionen som utför binärsökning på en array av heltal.
   ---***P-uppgift*** *(3p)*