Tentamen - Datastrukturer, algoritmer och programkonstruktion.

DVA104

Akademin för innovation, design och teknik

Onsdag 2014-01-15

Skrivtid: 08.30 – 13.30
Hjälpmedel: Inga
Lärare: Caroline Uppsäll
(kan nås på telefon 021-101456 (070-4616110)

Betygsgränser

Betyg 3:	25p (varav minst 6p P-uppgifter)
Betyg 4:	35p
Betyg 5:	42p
Max:	**46p (varav 12p är P-uppgifter)**

- 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!
- Uppgifter som är markerade som P-uppgifter måste lösas med kod i C (C# eller Ada). Övriga uppgifter behöver inte lösas med kod.
- Tips: Läs igenom hela tentan innan du börjar skriva för att veta hur du ska disponera din tid.
- Endast bonuspoäng som intjänats under instansen HT14 period 2 kan användas på denna tentamen.
- Intjänade bonuspoäng räknas ej till P-uppgifterna

Lycka till!

Uppgift 0

Läs instruktionerna på förstasidan noggrant. Följ dessa genom hela tentamen!

Uppgift 1 *(8p)*

a) Vad menas med divide & conquer?
b) Beskriv ett exempel på en algoritm som använder divide & conquer och på vilket sätt den använder paradigmen.
c) Vad är ett annat namn på en LIFO kö?
d) Varför vill du kunna mäta en algoritms komplexitet?
e) Vad menas med att en sorteringsalgoritm är naturlig?
f) Ge ett exempel på en sorteringsalgoritm som är in-place och som är naturlig.
g) Beskriv en funktion på en array som har linjär komplexitet – förklara också varför den har linjär komplexitet.
h) Varför är basfallet viktigt i rekursion?

Uppgift 2 *(7p)*

Antag att följande funktion (funk) är given i pseudokod och att `node` är en nod i en enkellänkad lista (tecknet ! betyder *skiljt från*).

```
node
{
      Integer data
      node next
}


funk(node, x)
      current = node
      while current != NULL do
          if current.data = x then
               return true
          end if
          current = current.next
      end while
      return false
```

a) Vad gör denna funktion exakt? (1p)
b) Implementera denna funktion i kod (2p) **P-uppgift**
c) Implementera en rekursiv version av funktionen som gör precis samma sak (4p) **P-uppgift**

Uppgift 3 *(3p)*

Du har ett uttryck som innehåller parenteser ’(’ och ’)’. Uttrycket kan också innehålla andra tecken.

Beskriv hur du kan med hjälp av nedanstående stack-ADT kontrollera om alla parenteser i uttrycket matchar. Alltså om varje ’(’ har en matchande ’)’. Du väljer själv om du vill beskriva detta genom att skriva kod, pseudokod, beskrivande text och/eller genom att rita. Din lösning ska på alla steg vara tydliga och enkla att följa.

Exempel på korrekt uttryck är: (x(x)(x(x(xxx))))
Exempel på icke korrekt uttryck är: (x)(x(xxx)

Den stack ADT du ska använda dig av kan lagra tecken (characters). Nedan finns dennes interface. Antag att stacken inte kan bli full.

Stacken är baserad på en länkad lista där varje nod håller datadel (character) och pekare till nästa nod (next)

```
Stack
{
     LinkedList *myStack
}
```

Precondition: stacken är inte full
Postcondition: c ligger högst upp på stacken
Returvärde: ingenting

```
Push(character c, stacken)
```

Precondition: det finns tecken på stacken
Postcondition: det översta tecknet på stacken är borttaget
Returvärde: ingenting

```
Pop(stacken)
```

Precondition: det finns tecken på stacken
Postcondition: stacken är oförändrad
Returvärde: det översta tecknet

```
character Peek(stacken)
```

Precondition: -
Postcondition: stacken är oförändrad
Returvärde: true om stacken är tom, annars false.

```
Boolean isEmpty(stacken)
```

Uppgift 4 *(7p)*

a) Beskriv hur Quicksort fungerar (1p)
b) Visa med bild/bilder (och text om det behövs) hur följande sekvens sorteras med hjälp av Quicksort. (3p)

5	8	3	4	9	1	6	7

c) Hur påverkar valet av pivot värde Quicksort algoritmens effektivitet? - diskutera (2p)
d) Vilken komplexitet har algoritmen i bästa fallet och varför? (1p)

Uppgift 5 *(8p)*

a) Beskriv hur en sekvens av data sorteras med hjälp av urvalssortering. (1p)
b) Skriv koden för urvalssortering (4p) **P-uppgift**
c) Vilken komplexitet har urvalssortering i bästa fall samt sämsta fall – vilka fall är det som är bäst och sämst? (2p)
d) Är algoritmen stabil? – motivera. (1p)

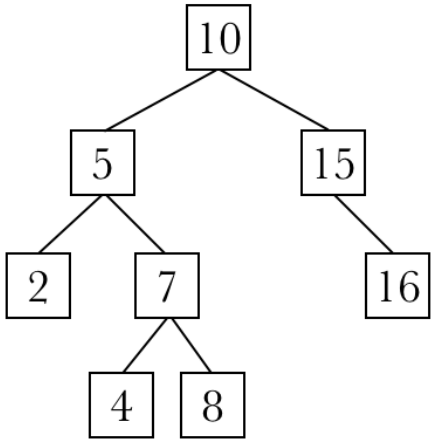
Uppgift 6 *(5p)*

Antag att du har en hashtabell med plats för 10 element- Hashfunktionen är $h(x) = (x \bmod 10)$ (tips: i praktiken blir detta alltså sista siffran i x).

a) Antag att tabellen använder chained-strategin (länkad lista) för att hantera krockar. Visa hur hashtabellen ser ut när följande element satts in i ordning: (2p)
   56 12 10 87 26 8 1 30 96 69 43
b) Beskriv hur sökning går till i en listad hashtabell (1p)
c) Antag nu istället att tabellen använder öppen adressering för att hantera krockar. Visa hur hashtabellen ser ut när samma sekvens som i a) sätts in (i samma ordning). (2p)

Uppgift 7 *(8p)*

Antag följande träd:



a) Är trädet ovan ett binärt sökträd? - motivera (1p)
b) Rita om trädet så att det blir ett balanserat binärt sökträd. (1p)
c) Antag följande träd och nod:

```
BTree
{
     node root
}
node
{
     integer data
     node leftChild
     node rightChild
}
```

Skriv koden för att skriva ut trädet i Preorder (rekursivt). (2p) **P-uppgift**

d) Vilken är komplexiteten i det bästa fallet för sökning i ett binärt sökträd? – motivera (1p)
e) Vilken är komplexiteten i det sämsta fallet för sökning i ett binärt sökträd? – motivera (1p)
f) Om man vill söka efter ett data i en stor mängd. När bör man använda hashtabell och när behöver man använda binärt sökträd – diskutera. (2p)