Tentamen 2D1321/2D1322 Tillämpad programmering och datalogi

Tisdag 20 december 2005 14-19. **Tillåtna hjälpmedel**: ***Pythonhäfte samt handskrivna rosa formelblad***
Resultat anslås senast 12 januari på Nadas anslagstavla. Max 100p, godkänt 50p. Bonus max 9p adderas.

1 Knepigt pyssel (20p)

Sudoku (数独) är ett pussel som går ut på att man ska placera ut siffror i ett rutmönster. Det klassiska, ursprungliga rutmönstret består av 3 × 3 rutor (”lådor”) som i sin tur består av 3 × 3 rutor. Det gäller att placera in siffrorna 1 till 9 på ett sådant sätt att varje vågrät rad, lodrät rad och låda på 3 × 3 rutor innehåller varje siffra exakt en gång.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

a) Beskriv kortfattat hur en djupet-först respektive en bredden-först algoritm fungerar i allmänhet.

b) Motivera vilken av de två algoritmerna som är att föredra för att lösa en sudoku?

c) Samma siffermönster kan uppträda flera gånger. Hur ska man hantera det? Förklara utförligt och beskriv de datastrukturer och metoder som behövs.

2 Knepiga pussel (10p)

Beta ska sortera ett antal pusselbitar i svårighetsordning. En del sorteringsalgoritmer fungerar inte alls för Beta därför att han har så kort närminne, han kan bara komma ihåg fyra saker samtidigt. Om han ska titta på fem pusselbitar så kommer han t.ex. att glömma bort den första när han kommer till den femte. Han har däremot inga som helst svårigheter med att avgöra om en pusselbit är svårare än en annan.

Beskriv en algoritm steg för steg hur Beta ska bära sig åt för att sortera pusselbitarna.

3 Kodknepande (12p)

Beta har lyckats lära sig en hel del programmering och äntligen fått till ett program som fungerar! Programmet skriver ut ett snyggt ’B’ på skärmen. Genom att konsekvent klippa och klistra i koden har Beta ett andra program som kan skriva ut ”BETA_BETALAR_ARBETE”. Programmet är mycket större än det första och innehåller inga egna funktioner överhuvudtaget.

a) Förklara för Beta flera fördelar med att införa funktioner i programmet.
b) Ge förslag på några funktioner.

4 KMP-knep (8p)

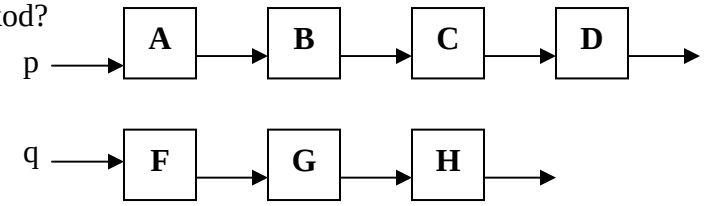
Beta är väldigt stolt över sitt program och är övertygad om att andra kommer att stjäla idén och vill därför ha hjälp med att skärskåda andras program. Tillverka en KMP-automat för att söka efter texten ”BETA_BETALAR_ARBETE”. Ange även next-vektorn.

5 knepiga listor (10p)

Beta försöker lära sig hantera länkade listor under lovet. Det är inte lätt. Givet följande listor.

a) Hur ser listan/listorna ut efter följande rader kod?

```
z = p.next
p.next = q.next
z.next.next = p
q = p = z
```



b) Hur skulle det ha sett ut om man istället skrivit följande rader kod?

```
p.next.next.next = q.next.next
q.next = p
p = q
```

6 rekursivt knåp (19p)

Beta har problem med en rekursiv listinsättningsfunktion.
a) Förklara för Beta vad som karaktäriserar en rekursiv funktion?

b) Vad är problemet med koden nedan?
c) Hur fungerar koden om man försöker anropa den ett par gånger?

```
def insert(p, value) :
    if (p == None): return Node(value)
    else: insert(p.next, value)
    return p
```

d) Skriv en fungerande listinsättningsfunktion.

7 knepigt med prioritering (13p)

Det är mycket som ska hinnas med innan julen. Plugga tentatal t.ex. För att underlätta har Beta prioriterat alla typtal och stoppat in dem i en prioritetskö. Bredden-först är viktigast enligt Beta.

20	Bredden-först
10	djupet-först
15	länkade listor
8	rekursion
7	huffman
13	KMP
14	Koduppprepning
2	Komplexitet
5	Prioritetsköer
6	funktionsanrop

a) Visa hur prioritetskön ser ut efter varje urtagning. Du behöver bara ange siffervektorn i varje steg [20, 10, 15, 8, 7, 13, 14, 2, 5, 6]

Om man har ungefär hundratusen element i en prioritetskö. Hur många operationer krävs det för att

b) Läsa av det mest prioriterade värdet (utan att plocka ut det).
c) Plocka ut det ur kön.

8 för att inte tala om komprimering (8p)

Huffmanträd kan användas för att komprimera text. Metoden är statistisk. Vad innebär det? När är den inte bra att använda?