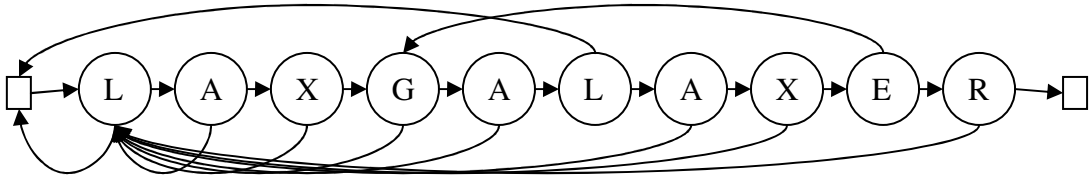
KTK CSC/CB, Mikael Lindahl

Lösningsskiss till tentamen i DD1320 Tillämpad datalogi 20090113
Uppgift 1

a)



b)
i – tillstånd

Next(i) – tillstånd att gå till vid fel bokstav

i	Next(i)	
1	0	L
2	1	A
3	1	X
4	1	G
5	1	A
6	0	L
7	1	A
8	1	X
9	4	E
10	1	R

Uppgift 2 n
Astrogenius: $n + n\log_2(n) = n(1+ \log_2(n))$
Stellatype: n^2/100 $= n(n/100)$

Prestandatabell:

	Stellatype	Astrogenius	Snabbast
n	n/100	$1+ \log_2(n))$	Stellatype
2	0.02	2	Stellatype
4	0.04	3	Stellatype
8	0.08	4	Stellatype
16	0.16	5	Stellatype
32	0.32	6	Stellatype
64	0.64	7	Stellatype
128	1.28	8	Stellatype
256	2.56	9	Stellatype
512	5.12	10	Stellatype
1024	10.24	11	Stellatype
2048	20.48	12	Astrogenius

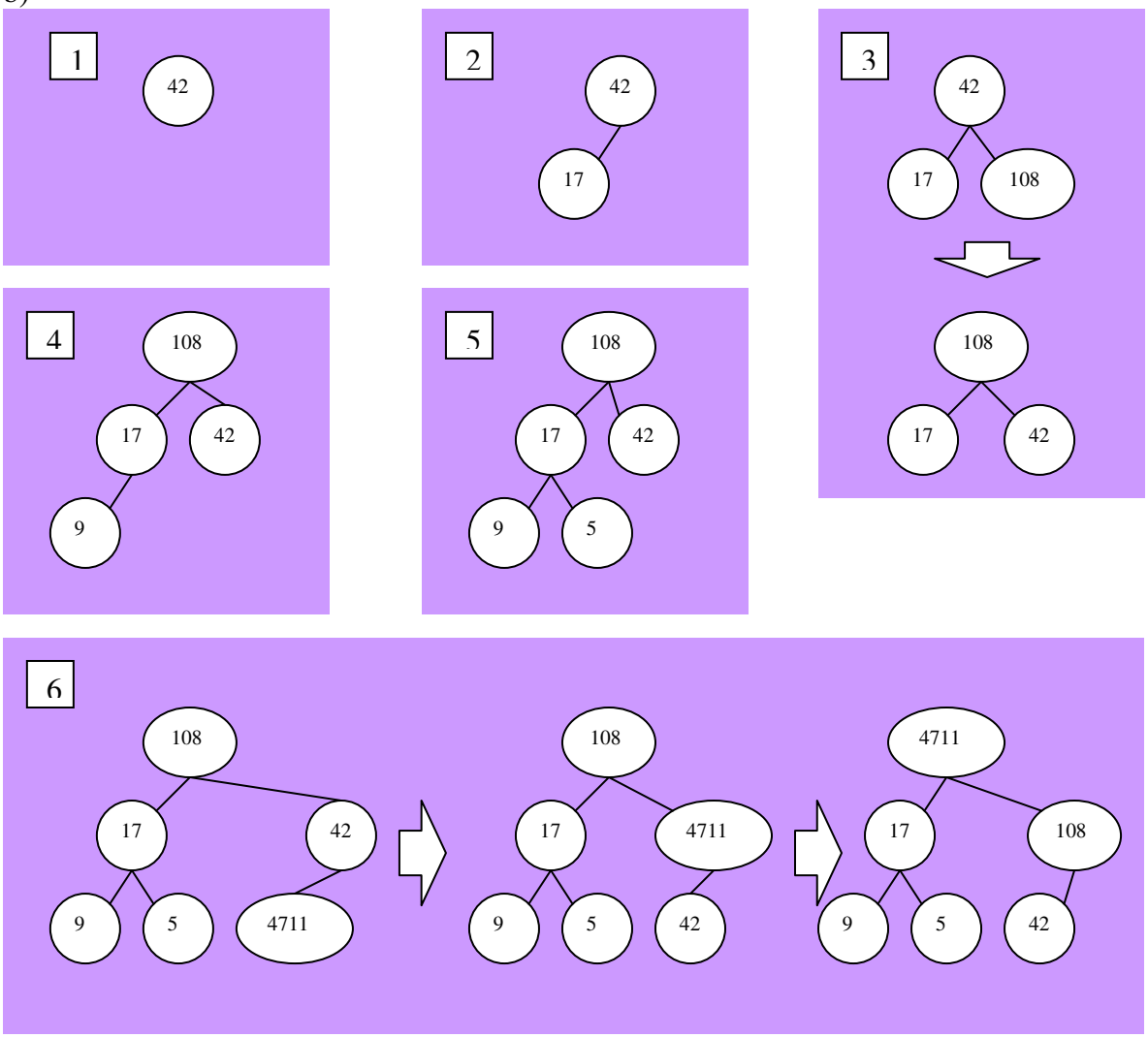
Om antal stjärnor som ska klassas är mindre än 1024 köp stellatype. Om fler köp Astrogenius.

Uppgift 3

a)
Insättning: 0 – Lägg in sist i heap
1 – Byt med förälder om inte heapvillkor uppfyllt.
2 – Upprepa 1 tills heapvillkor uppfyllt

n – antal element i heapen. Vid insättning kan max $\log_2(n)$ byten mellan förälder och barn göras innan vi kommer till första elementet i heapen.

b)



c) Heapvilkor förälder > barn (maxheap)
4711 > 17 och 108, 17 > 9 och 5, 108 > 42

Uppgift 4
a) LPNFUSFB ska dekrypteras. Transpositionschiffer, Caesar/rot13 eller RSA? Vi provar transposition genom att dela upp och försöka läsa kolumnerna:

LP	LPN	LPNF	LPNFU	LPNFUS	LPNFUSF	Det gav inget läsbart.
NF	FUS	USFB	SFB	FB	B	
US	FB					

KTK CSC/CB, Mikael Lindahl

FB

Caesar då? Med ett program skulle vi kunna prova alla varianter:
MQOGVTGC, NRPHWUHD osv
Men det är jobbigt att göra för hand. Vi kollar istället vilken bokstav som är vanligast (här: F). Och byter den mot E, vilket ger KOMETREA

b)
Caesar med +25 eller -1

Med långa meddelanden går det att beräkna bokstavsfrekvens. Det plus tillgång till en frekvenstabell för det aktuella språket gör det enkelt att knäcka krypteringen. (I både svenska och engelska är E den vanligaste bokstaven.)

Uppgift 5
a)
Abstrakt datatyp definieras av vilka operationer som det går att göra på datatypen, dvs ett gränssnitt.

Skapa en klass med metoder som motsvarar abstrakta datatypens operationer. Klassen kapslar in metoderna dvs. det behövs ingen kunskap om hur metoderna är gjorda.

b)
Litet antal värden -> hastigheten spelar ingen roll. Använd metod som är lättast att skriva.

c)
Nej, t.ex. vid Lempel-Ziv komprimering används ingen frekvenstabell

d)
Ja, det skapas då nya parametrar vid varje rekursivt anrop

e)
(1) Hashtabell med få krockar O(1)
(2) Hashtabell med många krockar O(n)
Fall 1 då är hashtabell snabbare (O(1)<O(log(n))
Fall 2 då är hashtabell långsammare (O(n)>O(log(n))

Uppgift 6

a)

<sök> ::= inget | <ord> | <bygg> <AND> <bygg> | <bygg> <OR> <bygg> | <NOT> <bygg>
<bygg>::= <ord> | (<bygg> <AND> <bygg>) | (<bygg> <OR> <bygg>) | <NOT> <bygg>
<ord>::= energi | stjärna | supernova | idol | etc…
<AND> ::= AND
<OR> ::= OR
<NOT> ::= NOT

Alternativt slå ihop <sök> & <bygg>:
<sök> ::= inget | <ord> | <sök> <AND> <sök> | <sök> <OR> <sök> | <NOT> <sök>|(<sök>)

b)
(energi AND (stjärna OR supernova)) AND NOT idol
(<sök>)
<sök> AND <sök>
<ord>
energi

Och så vidare…

Uppgift 7
Breddenförstsökning med kö är bäst:

1. Lägg stamfadern i kön.
2. Ta ut det första objektet ur kön.
3. Skapa alla dess barn och lägg in dom i kön.
4. Om någon av barnen är lösningen så är vi klara. Annars - upprepa från punkt 2.
5. När lösningen hittas, följ faderspekarna och skriv ut kedjan.

En position är en lista med hinkarnas innehåll, den lagras i objektet.
Problemträdet startar med två tomma hinkar. Ett barn skapas genom påfyllning från sjön eller hällning från hink i till hink j.

0,0
2,0 0,5
0,2 2,5 2,5 2,3 osv

För att lätt kunna jämföra när man letar dumbarn kan man koda en lösningsposition som en sträng. I python konverterar du enkelt en lista med tal till en sträng genom att skriva str(LISTNAMN).

Det går ocksp att göra om till ett heltal, till exempel med
$10000h_1+100h_2+h_3$, eller
$4096h_1+64h_2+h_3$,
där h_i är innehållet i hink i.
För varje position man skapar sparar man koden i ett dumbarnsbinärträd eller en dumbarnshashvektor.

När en lösning dyker upp skriver man ut kedjan rekursivt, för att få den i rätt ordning.

Klasserna Queue och Bintree behövs och i huvudmodulen finns funktionerna makeChildren och writeChain och en nodtyp med hinklista och faderspekare.