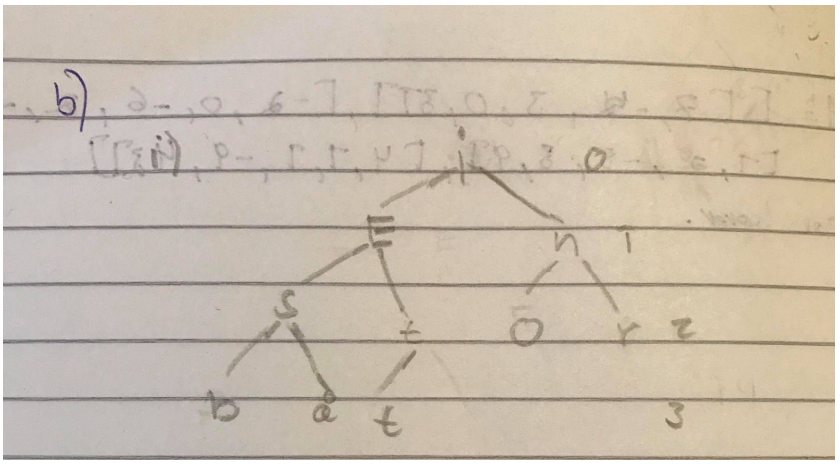
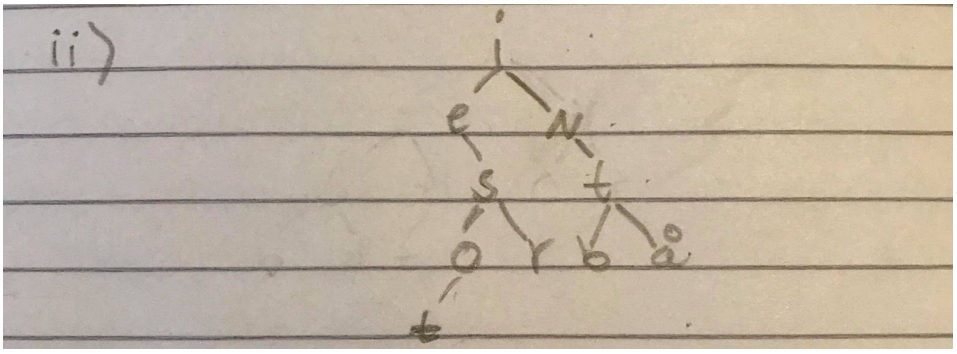
Oppgave 1

a) Et binært tre er en samling av noder som danner et på som har minst to barn. Altså hver node må ha minst 2 barn.
b) Høyden til et tre er lengden til den lengste stien fra roten til siste bladet.
c) I et fullt tre er alle blad på samme nivå, alle noder er enten barn eller har de nøyaktig like barn. Et komplett er et tre som enten er eller fullt til nest siste nivå. på siste nivå er bladnodene plassert lengst til venstre så mulig.
d)

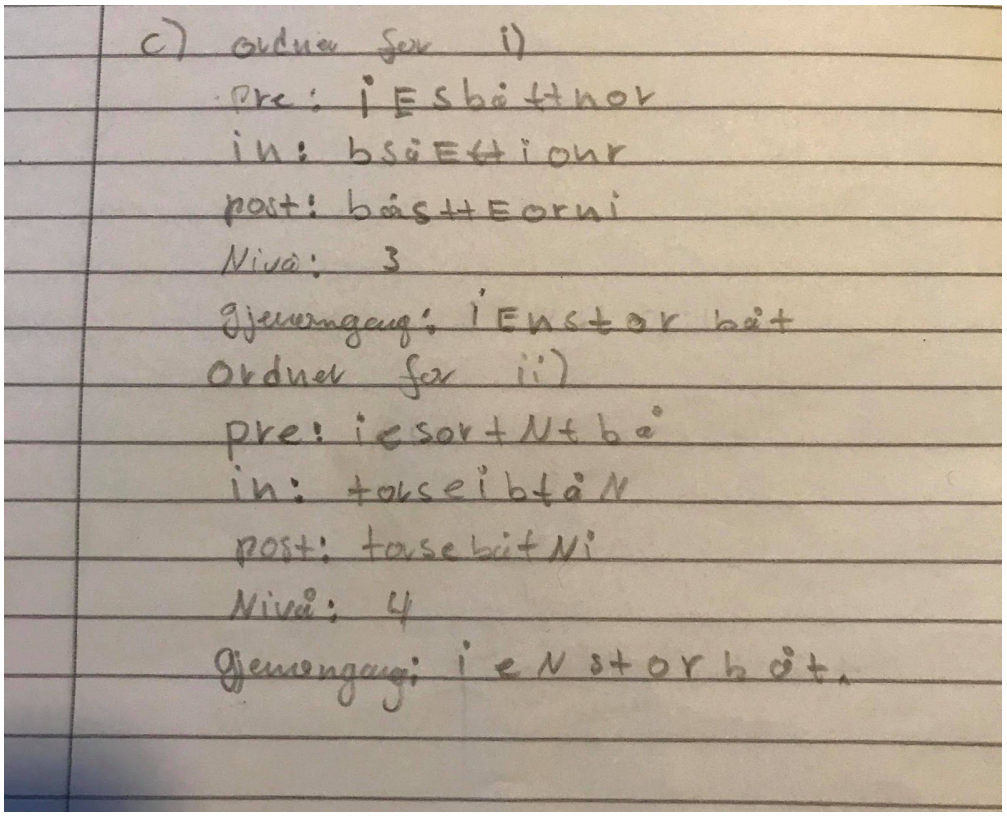
i)



II)



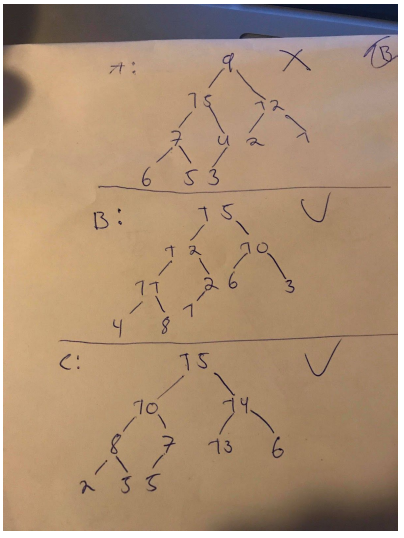
iii)



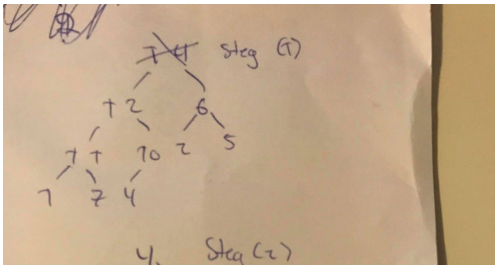
Oppgave 3

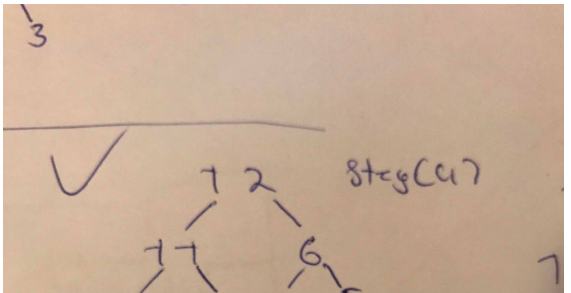
a) Et binært haug er en abstrakt datastruktur, det brukes mest til å sortere data eller lage prioriterings køer. vi har to ulike hauger den første er minmumhaug så for hvert intern node der elementet er mindre eller lik venstre og høyre barn så er det en minimumshaug. Men for maksimumhaug så er det motsatt der elementet må være større eller lik venstre og høyre barn.

b) B og C er maksimum hauger, mens A ikke er det på grunn av første elementet (9) i nivå 0 er mindre enn barn.



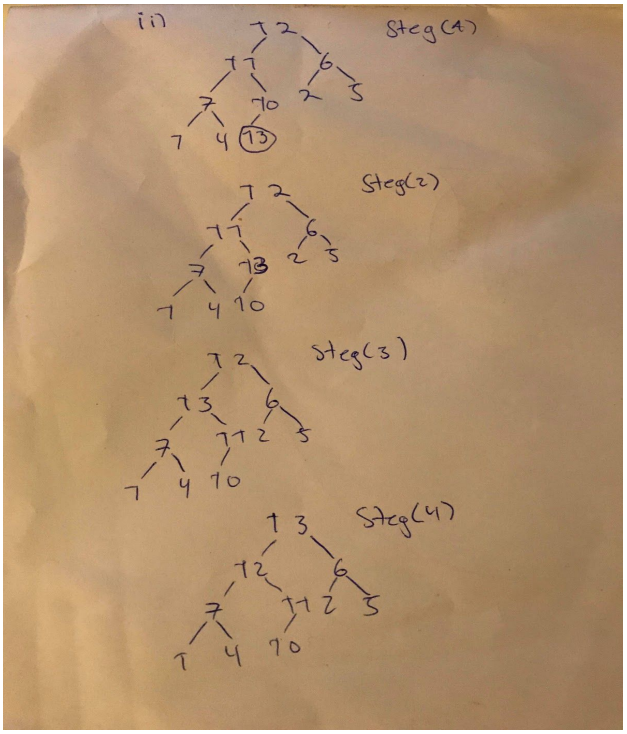
c)





i)

ii)

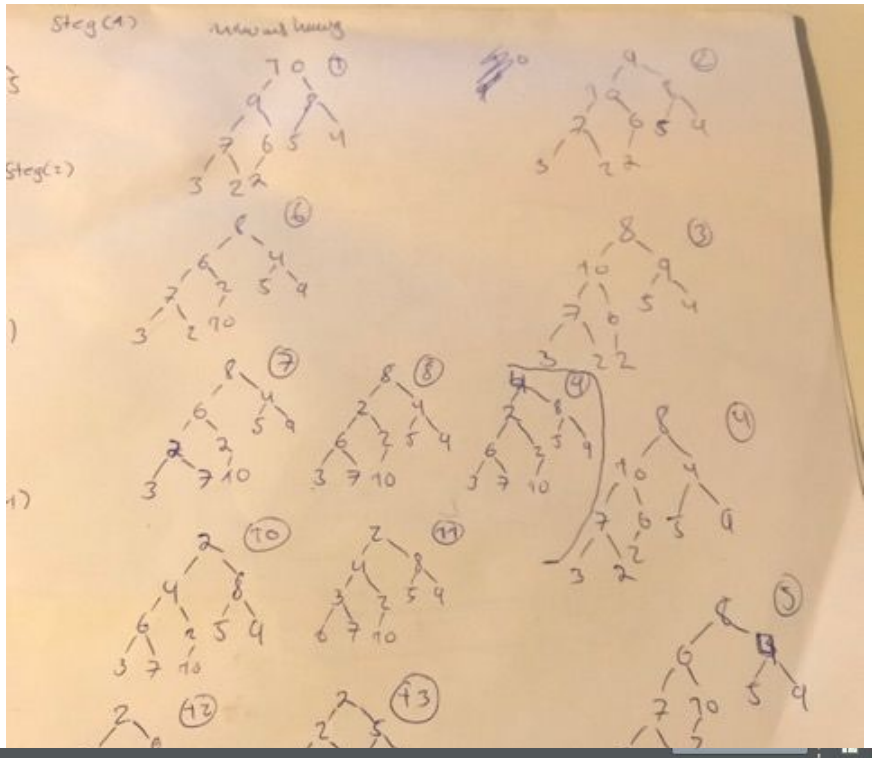


d) Sortering på en maks haug er å legge til hvert element i listen til haugen og så fjerne det en om gangen. men siden det er en makshaug da er resultat avtagende.

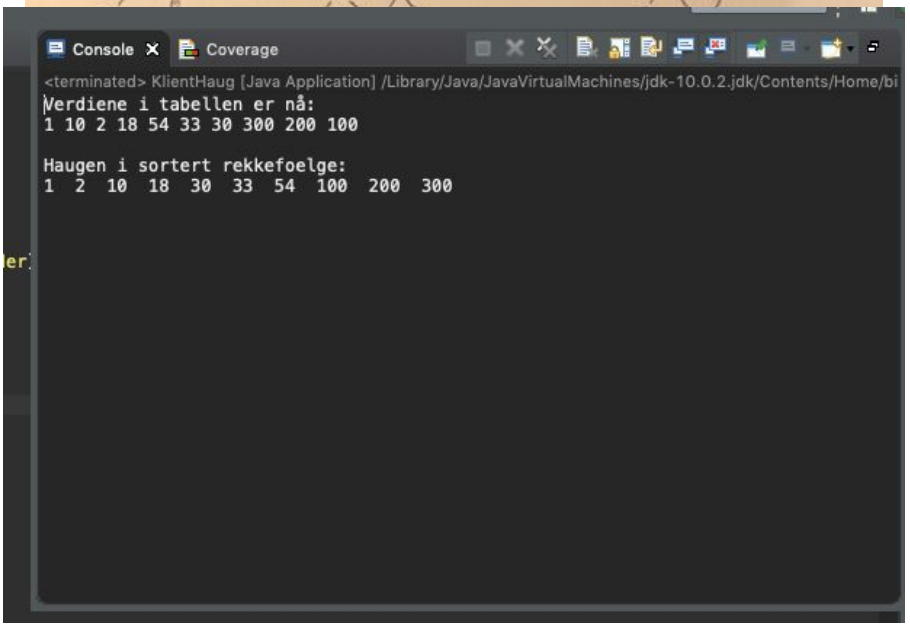
e) Hvis vi vet posisjons på foreldrene og deres barn så kan vi sammenligne den første noden med barna, så bytte hvis nødvendig. vi går gjennom tabellen baklengs til vi når roten,

det får å lage to sammenligninger da er det O(n) for bygge opp haugen. For fjerning av element i haugen og vedlikeholde haugen så er det O(n log n) og dette er effektivt enn 2 * n + n log n.

f)



g)



Oppgave 4)

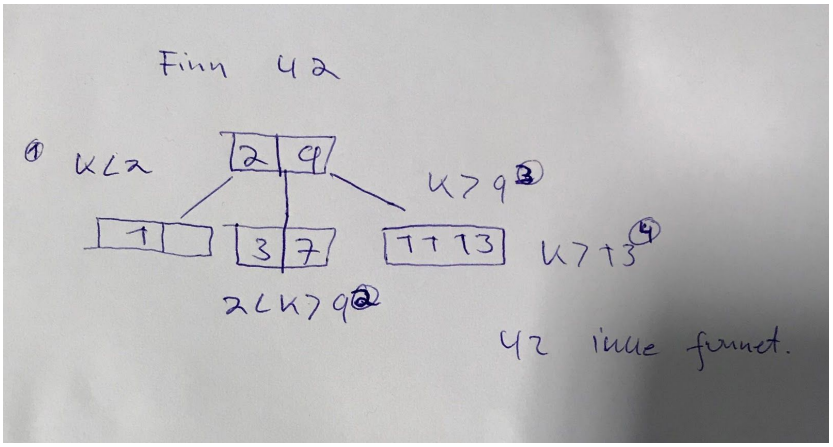
a) 2-3 tre er en flervalg søketre der hver node har enten null, 2 eller 3 barn.Det brukes mest til å finne et element.

b) En node med null eller 2 barn kalles for 2-node. Det inneholder et element og det har enten ingen eller 2 barn. Elementene i venstre undertre enn mindre enn elementen i 2-Noden. og i høyre så er det større. Den er prikk lik som en BS-tre.

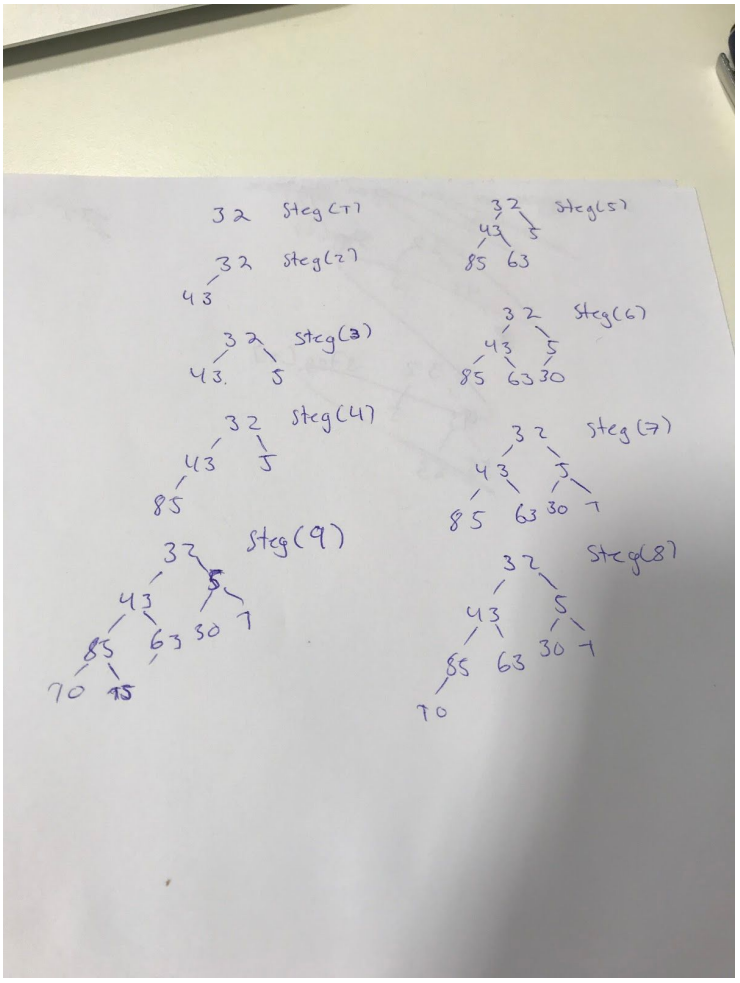
En node med null eller 3 barn kalles for 3-node og det inneholder 2 elementer ett som er det minste og ett som er det største. Den har enten ingen barn eller 3 barn.

c)

i)

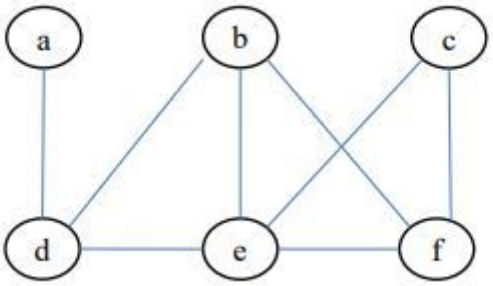


ii)



Oppgave 5

a)



Bruker c som startnode

Kø	Besøkt
c	c
ebdf	ce
bdf	ceb
dfa	cebd
fa	cebdf
a	cebdfa

b)

Bruker c som startnode

Kø	Besøkt
c	c
ebdf	cf

ebd	cfd
eba	cfda
eb	cfdab
e	cfdabe

c)
a:(a,d)
b:(b,d),(b,e),(b,f)
c:(c,f),(c,e)
d:(d,a),(d,b),(d,e)
e:(e,b),(e,c),(e,d),(e,f)
f:(f,b),(f,c),(f,e)

Nabomatrise:

	a	b	c	d	e	f
a				x		
b				x	x	x
c					x	x
d	x	x			x	
e		x	x	x	x	
f		x	x		x	

Naboliste
a:d
b:d→ e→ f
c:e→ f
d:a→ b→ e
e:b→ c→ d→ f
f:b→ c→ e
Oppgave 6

a)
Hash-funksjon:
Hash som bruker første bokstav i navnet for å bestemme posisjon i tabellen(bruker det engelske alfabetet)

Kollisjon:
Kollisjon har vi når to ulike nøkler i en tabell referer til samme celle(posisjon)

Clustering:
Det er når hash-addressene klynger seg sammen(det er større sjanse for å finne en hash-addresse når flere er rundt). Dette er noe vi finner spesielt i lineær probing.
b)

SU65431
TA14374
ZX87181
ST47007
VV50000
UV14544
SU32944

c)
Hascodes:
ab. 97*31^1+98*31^0 = 3105
123. 49*31^2 + 50*31^1+51*31^0 = 48 690

d)

Fordi objektene gir ut en referense og det er det man tar hasCode() på. Her trenger man istedenfor å sjekke på innhold med en egen equals metode i Student, og lage en if setning i main-metoden.

e)
Søking gjennom hashCode() er vesentlig raskere enn binærsøking.