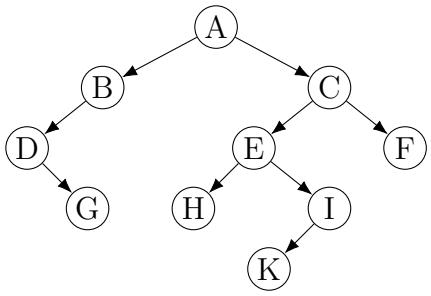
Het tentamen bevat vier opgaven. Graag elke opgave op een nieuwe pagina beginnen.
Pseudo-code mag do/od gebruiken of er meer als C++ uitzien, dat is niet belangrijk.
Geef steeds voldoende uitleg. Zie ook het toegevoegde commentaar. Succes.

1. **a)** Neem onderstaande boom over, en breng symmetrische bedrading aan (alleen rechter draden).



We bekijken de volgende abstracte manier om een inorde (symmetrische) wandeling door een boom uit te voeren.

inorde
```
move to first node
repeat
   visit node
   move to successor node
until done
```

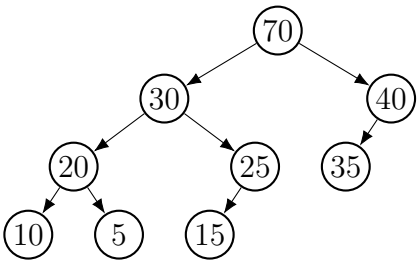
b) Leg uit hoe de opdrachten *move to first*, *move to successor* en *done* worden geïmplementeerd in een bedrade binaire boom. Gebruik pseudo code.

Het is niet voldoende om alleen een wandelalgoritme te schrijven. Voor `move to successor` moet duidelijk zijn hoe we vanuit een (willekeurige) knoop de opvolger kunnen bereiken. Gebruik geen recursie, geen stapel, maar natuurlijk wel de draden.

c) Idem, nu in een onbedrade boom, maar gebruikmakend van een stapel.

2. **a)** Beschrijf *leftist trees*, met hun basisoperatie 'ritsen' (*Zip*).

b) Hieronder een max-heap, geïmplementeerd als leftist tree.



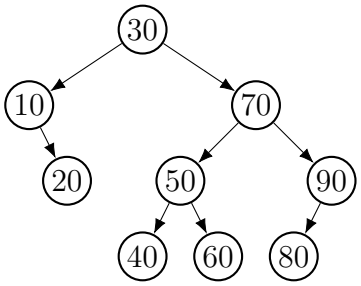
(i) Voeg 60 toe aan de max-heap.
(ii) Verwijder het maximum uit de (oorspronkelijke) heap.

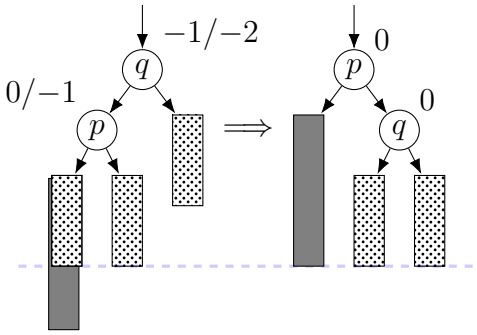
c) **(i)** Beredeneer dat een leftist tree waarvan de wortel *nil path length* k heeft, ten minste $2^k - 1$ knopen heeft.

(ii) Wat kun je zeggen over het maximaal aantal knopen in die boom?

3. **a)** Beschouw de volgende AVL boom A (hieronder, links).

Welke rotatie wordt uitgevoerd (waar, richting, enkel/dubbel) als we sleutel 15, 35, 85, respectievelijk 95 toevoegen? (Dus steeds één sleutel ten opzichte van de oorspronkelijke boom A. Je hoeft niet de resulterende bomen te tekenen.)





b) Verwijder de waarde 30 uit boom A. Dit kan op twee manieren, laat deze beide zien, met resultaat.

c) Plaatje hierboven rechts komt uit het diktaat, de *LL-case* bij rotatie in een AVL-boom: de subboom links-links wordt langer, knoop q raakt uit onbalans. De illustratie is voor het geval dat p oorspronkelijk balans 0 heeft.

Moeten we bij het implementeren van de *LL-case* ook andere balans van p meenemen? Leg uit.

d) Een gebalanceerde boom maakt zoeken efficiënt. Geef een toepassing waar je toch een lineaire lijst zou gebruiken om elementen op te slaan.

4. **a)** Wat is de tijdscomplexiteit (O) van de volgende operaties?

(i) Naïef zoeken naar een patroon P in de tekst T

(ii) Knuth-Morris-Pratt (KMP) patroon opstellen en zoeken.

b) Bepaal de *failure-links* voor het patroon P = `1201 2120 12`.
(Spaties voor de leesbaarheid).

Geef aan hoe de failure links tijdens het opstellen worden gebruikt.

c) We zoeken naar P in de tekst T = `1201 2012 2120 1201`
Geef nauwkeurig aan hoe het zoeken volgens de KMP-methode gebeurt.
Welke letters worden telkens met elkaar vergeleken?

Waar worden de failure links gebruikt?