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. Succes.

1. We kijken naar een symmetrisch(=in-orde) bedrade boom (*inorder threads*) met alleen draden naar rechts. (De draden zijn permanent, en worden van takken onderscheiden door een boolean `Isthread` in elke knoop.)

We onderzoeken in deze opgave welke boomwandelingen we kunnen maken met symmetrische bedrading. Gebruik in de antwoorden pseudo-code en een plaatje.

a) – Hoe vinden we de eerste knoop in in-orde volgorde uitgaande van de wortel `R`?
– Hoe vinden we de in-orde opvolger van een knoop `curr`?

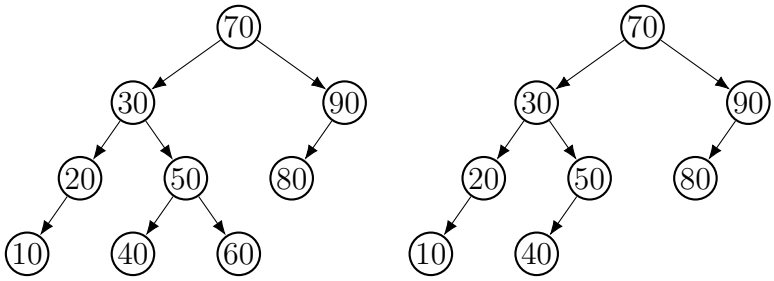
De volgende onderdelen kijken naar andere boomwandelingen, maar nog steeds in een symmetrisch bedrade boom!

b) – Hoe vinden we de eerste knoop in pre-orde volgorde uitgaande van de wortel `R`?
– Hoe vinden we de pre-orde opvolger van een knoop `curr`?

c) – Hoe vinden we de eerste knoop in post-orde volgorde uitgaande van de wortel `R`?
– Laat met een eenvoudig voorbeeld zien dat we *niet* van elke knoop `curr` de post-orde opvolger kunnen bereiken vanuit `curr` als we alleen takken en draden mogen volgen.

2. **a)** Geef de definitie van AVL-boom.

b) Beschouw de volgende AVL boom A (hieronder links). We gaan één sleutel toevoegen. Welke rotaties worden uitgevoerd (waar, richting, enkel/dubbel) als die sleutel gelijk is aan respectievelijk 5, 35, 65 en 95? (Dus steeds ten opzichte van de oorspronkelijke boom A. Je hoeft niet de resulterende bomen te tekenen.)



c) Geef twee redenen waarom verwijderen van een waarde uit een AVL boom ingewikkelder is dan het toevoegen. (Een reden geldt voor alle binaire bomen, de andere is AVL specifiek.)

Boom B (hierboven rechts) heeft één knoop minder dan boom A.

d) Verwijder de waarde 70 uit boom B. Dit kan op twee manieren. Laat deze beide zien.

3. **a)** Geef de definitie van *leftist trees*, en beschrijf hun basisoperatie 'ritsen' (*Zip*).

b) Leg uit hoe leftist trees gebruikt kunnen worden als implementatie van de abstracte datastructuur *Priority Queue*.

c) We gaan uit van een leftist tree waarvan de wortel *nil path length* k heeft.

(i) Wat kun je zeggen van de lengte van het pad van de wortel naar rechts?

(ii) Beredeneer dat de boom ten minste $2^k - 1$ knopen heeft.

(iii) Leg uit dat er geen maximum op het aantal knopen gegeven kan worden (als we alleen k weten).

4. **a)** Geef een patroon (lengte m) en tekst (lengte n) waarvoor het naïeve zoekalgoritme slecht presteert. Wat is de complexiteit?

De methode van *Knuth-Morris-Pratt* wordt gebruikt om een patroon P[1..m] in een tekst T[1..Lengte] te zoeken. Daartoe worden *failure-links* opgesteld.

b) Geef een efficiënt algoritme om de failure links op te stellen.

c) Bepaal de *failure-links* voor het patroon P = `ABCABABC`.

d) We zoeken naar P in de tekst T = `ABCA BCAB AABC ABAB BABC` (hier staan de spaties voor de leesbaarheid). Geef nauwkeurig aan hoe het zoeken volgens de KMP-methode gebeurt. Welke letters worden telkens met elkaar vergeleken?