Het tentamen bevat ditmaal zes *opgaven van twee onderdelen, zodat een betere spreiding van de onderwerpen ontstaat. Gevraagde functies en programma's mogen in pseudo-code gegeven worden. Geef steeds voldoende uitleg.*

1. **a)** Wat is een abstracte datastructuur (ADT) ?

b) Beschrijf de ADT *stapel* (=*stack*) en schets twee geschikte implementaties.

2. *Wandelingen op binaire bomen.*

Onderstaande algoritmes in pseudo-code voeren een pre-orde respectievelijk een in-orde (symmetrische) wandeling uit gebruik makend van een stapel.

pre-order

```
iterative-preorder( root )
    S.create()           // stack
    S.push( root )
    while ( not S.isEmpty() )
    do
        node = S.pop()
        if (node != nil)
        then
            visit( node ) // pre-order
            S.push( node.right )
            S.push( node.left )
        fi
    do
end // iterative-preorder
```

in-order

```
iterative-inorder( root )
    S.create()
    node = root
    // move to first node (left-most)
    while (node != nil)
    do
        S.push( node )
        node = node.left
    od
    while ( not S.isEmpty() )
    do
        node = S.pop()
        visit( node )    // inorder
        node = node.right
        while (node != nil)
        do
            S.push( node )
            node = node.left
        od
    do
end // iterative-inorder
```

a) Leg uit welke knopen er op de stapel staan gedurende elk van deze wandelingen.

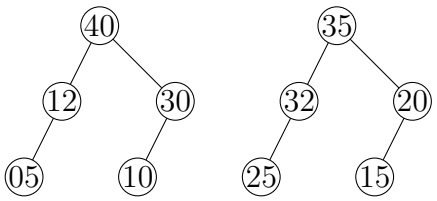
b) Er is een 'generiek' boom-wandel algoritme waarin elke knoop driemaal wordt bezocht (zoals we uitgewerkt hebben bij link-omkering). Maak daarvoor de volgende tabel af, met de globale bezoek-teller `visit`.

`visit`	node-test	direction	new `visit`
1	...	...	...
...	...	...	...

(Alleen de tabel is voldoende, met enige uitleg.)

3. Leftist trees vormen de implementatie van de ADT priority queue, met 'ritsen' als basis-operatie (*zip*).

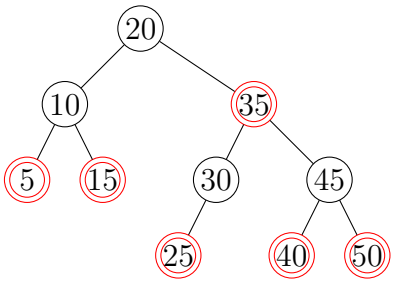
a) Rits de onderstaande leftist trees, zodat er weer een leftist tree ontstaat.



b) Beschrijf efficiënte implementaties voor de priority queue operaties Insert, DeleteMax, en IncreaseKey met behulp van ritsen en leftist trees.

4. **a)** Wat zijn de definiërende eigenschappen van de rood-zwart boom (*red-black tree*)?

b) Gegeven is de volgende rood-zwart boom, waarbij 'rode' knopen een extra cirkel hebben gekregen.



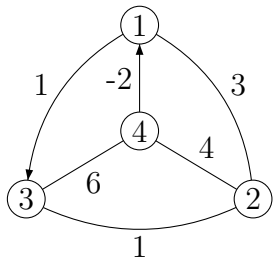
Welke boom ontstaat als we hieraan toevoegen

i. eerst 55 en vervolgens 7.

ii. eerst 7 en vervolgens 55.

Benoem de achtereenvolgende operaties en geef relevante tussenresultaten.

5. Onderstaande graaf heeft negatieve gewichten (en is slechts gedeeltelijk gericht).



a) Laat zien dat het algoritme van Dijkstra *niet* toepasbaar is op grafen met negatieve gewichten, door het algoritme toe te passen op bovenstaande graaf. Kies een geschikte beginknoop.

Het algoritme van Floyd (*all pair distances*) is wel geschikt voor negatieve gewichten, zolang er geen negatieve kringen zijn in de graaf.

b) Geef het algoritme van Floyd.

6. Ziv-Lempel-Welsh codering.

Het alfabet heeft vier letters `a`, `b`, `c`, `d`. Letter `a` krijgt code 1.
Geef telkens welke letters 'geleerd' worden, en de uiteindelijke codeboom.

a) Codeer `cdbc acac daaa bc`, spaties staan hier alleen voor de leesbaarheid.

b) Decodeer 2 4 4 5 1 3 7 11 1 2 14 10.