Dit 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. **a)** Onderstaande bekende algoritmes voeren een pre-orde respectievelijk een in-orde (symmetrische) wandeling uit, gebruik makend van een stapel.

```
iterative-preorder( root )
    S : Stack
    S.create()
    S.push( root )
    while ( not S.isEmpty() )
    do  node = S.pop()
        while (node != nil)
        do  visit( node ) // pre-order
            S.push( node.right )
            node = node.left
        od
    od
end // iterative-preorder
```

```
iterative-inorder( root )
    S : Stack
    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
    od
end // iterative-inorder
```

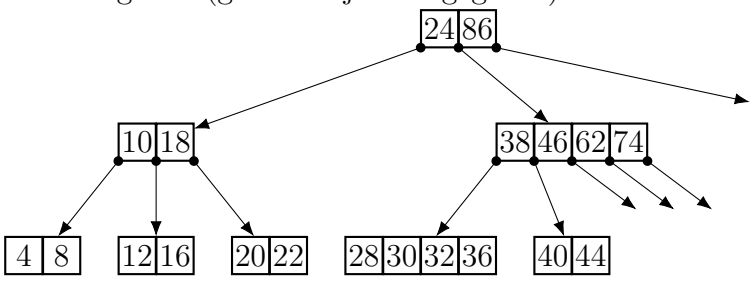
Tijdens elk van deze wandelingen worden knopen van de boom tijdelijk op een stapel gezet. Leg uit welke knopen van de boom er op de stapel staan op het moment dat een bepaalde knoop `node` bezocht wordt.

b) Ga nu uit van een boom-representatie waarin elke knoop naast de standaard `left`- en `right`-pointer ook een `parent`-pointer heeft. Bovendien is er een veld `isLeftChild` dat aangeeft of de knoop een linkerkind is.

Geef een passend algoritme voor de post-orde wandeling. Zonder stapel of recursie!

2. **a)** Op welke manier wordt, in de definitie van een B-boom, de wortel anders behandeld dan de overige knopen? Waarom is dat?

Beschouw de volgende (gedeeltelijk weergegeven) B-boom B van orde 5:



b) i. Verwijder sleutel 18 uit boom B.

ii. Voeg nu sleutel 18 weer toe aan de ontstane boom.

Zorg ervoor dat we bij het toevoegen en verwijderen steeds een B-boom van orde 5 behouden. Geef een korte toelichting en (zo nodig) tussenresultaten. Ongewijzigde delen van de boom kunnen schematisch worden aangegeven.

c) We kijken naar een B-boom van orde m. Bij toevoegen van een sleutel moet soms een knoop gesplitst worden. Hoeveel sleutels heeft de knoop op dat moment? En hoeveel sleutels hebben de twee nieuwe knopen?

3. We gaan in deze opgave uit van max-heaps.

a) Geef de definitie van een *binary heap*, en beschrijf de twee basis-operaties *bubble-up* en *trickle down*.

b) Illustreer hoe de priority queue operaties *DeleteMax* en *Insert(*50*)* worden uitgevoerd, aan de hand van de volgende heap: $[90, 75, 80, 70, 55, 40, 60, 15, 45, 10, 30, 20]$.

c) Van een ongeordend array kan op efficiente wijze een heap worden gemaakt. Illustreer hoe dit werkt aan de hand van het volgende array: $[50, 10, 30, 90, 20, 40, 70, 80, 60]$.

4. Beschouw hashen in een zgn. 'open' hashtabel met twee hash-functies h en p. Het $i+1$-ste bezochte adres $h(K, i)$ is zoals gewoonlijk $h(K) - i \cdot p(K)$ (modulo M).

a) Welke twee soorten clustering onderscheiden we bij hashen met open adressering? Geef een korte beschrijving.

b) In een tabel `T[0..10]`, dus $M = 11$, worden achtereenvolgens de sleutels 16, 29, 36, 40, 9, 20, 28 geplaatst, met adresfunctie $h(K) = K \bmod 11$ en lineair hashen (stapgrootte 1).
(i) Laat zien welke tabel ontstaat, maar geef op een overzichtelijke manier ook alle plekken waar de sleutels geprobeerd worden.
(ii) Hoeveel stappen kost het om te zien of 18 in T opgeslagen is?

c) Idem, nu met een stapfunctie $p(K) = 1 + (K \bmod 10)$.