Het tentamen bevat vier opgaven. Graag elke opgave op een nieuwe pagina beginnen. Geef steeds voldoende uitleg. Succes.

1. **a)** Beschrijf de abstracte datastructuur *Stapel* (=*Stack*): wat wordt opgeslagen, wat zijn de standaard operaties (en wat doen ze)? Schets twee bekende implementaties.

b) Als we vaak de k-de sleutel (in grootte, dwz. in inorde) in een binaire zoekboom willen vinden (voor een variabele k) hoe kan de boom dan worden aangepast?

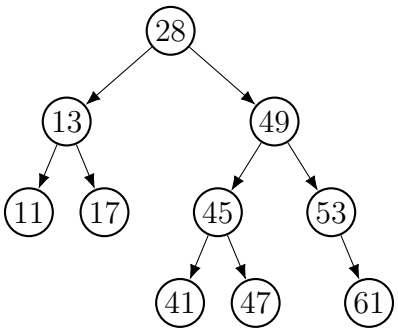
Wat verandert er aan de (aangepaste boom) wanneer een sleutel wordt toegevoegd?

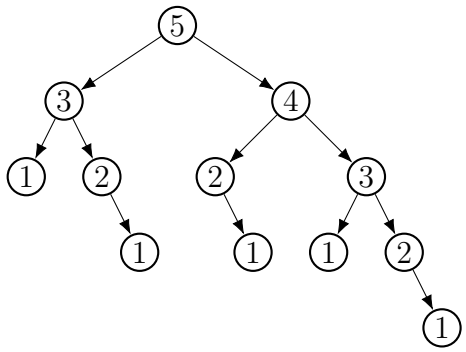
c) Bij een *splay tree* worden sleutels nadat ze gevonden zijn 'omhoog geroteerd'.

(i) Beschrijf de benodigde operaties: *zig-zag* en *zig-zig*.

(ii) Teken een lineaire boom van zeven knopen, met alleen rechter kinderen. Pas de splay operaties toe op de laagste sleutel in de boom.

2. *Over AVL-bomen.* Deze twee bomen worden bij de volgende onderdelen gebruikt.





a) Neem aan dat we een AVL-boom met balans-factoren gegeven hebben. We voegen een nieuwe knoop aan de boom toe, en het blijkt dat een rotatie nodig is. Beschrijf de plek in de boom waar die rotatie plaats vindt.

b) Voer de volgende opdrachten steeds uit voor de boom links.

(i) Welke boom onstaat als we knoop 34 toevoegen.

(ii) Idem voor knoop 56.

(iii) Welke AVL-boom ontstaat als we knoop 28 verwijderen?

Geef uitleg en tussenstappen: welke rotaties vinden waar plaats?

c) Hierboven (rechts) de structuur van een minimale AVL-boom van hoogte 5, dwz. met zo weinig mogelijk knopen. De hoogte wordt gemeten in het aantal knopen.

(i) Beredeneer zien dat elke AVL-boom van hoogte h tenminste $\lceil \frac{h}{2} \rceil$ geheel gevulde nivo's heeft.

(ii) Nu we dit weten, hoeveel knopen heeft een AVL-boom van hoogte h dus ten minste?

3. Ga in deze opgave uit van max-heaps.

a) Beschrijf de abstracte datastructuur *priority queue*: wat wordt opgeslagen, wat zijn de standaard operaties (en wat doen ze)?

b) Geef de definitie van een *binary heap*. Beschrijf hoe de twee basis-operaties *bubble up* en *trickle down* gebruikt worden om een priority queue te implementeren.

c) De heap-structuur is gemotiveerd door sorteren: *heap-sort*. Bekijk het gedeeltelijk gesorteerde array $[\,8, 5, 7, 4, 2, 6, 1, 3, 9, 10, 11, 12\,]$ met twaalf elementen. De laatste vier elementen zijn reeds gesorteerd.

(i) Welke eigenschap hebben de eerste acht elementen? Controleer dit.

(ii) Breng stapsgewijs twee extra elementen over naar het gesorteerde gedeelte, en hou de overige elementen weer op orde. Leg uit wat u doet.

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 punten zijn belangrijk bij de keuze van de adresfunctie h ?
Waarop moeten we letten bij de keuze van de stapfunctie (*probe function*) p?

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

c) In een tabel `T[0..10]`, dus $M = 11$, worden achtereenvolgens de sleutels 10, 22, 31, 4, 15, 28, 17, 88, 59 geplaatst, met adresfunctie $h(K) = K$ mod 11 en lineair hashen (stapgrootte 1).
Laat zien welke tabel ontstaat, maar geef op een overzichtelijke manier ook alle plekken waar de sleutels geprobeerd worden.