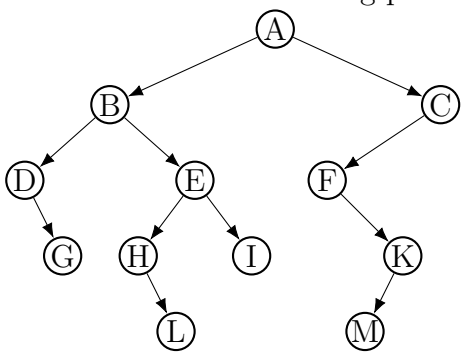
Dit tentamen bevat vier opgaven, elk voor $2\frac{1}{2}$ punt. Succes!

Geef steeds voldoende uitleg. Graag elke opgave op een nieuwe pagina beginnen.
Pseudo-code mag do/od gebruiken of er meer als C++ uitzien, dat is niet belangrijk.

1. We kijken naar een symmetrisch(=inorde) bedrade boom (*inorder threads*) met alleen draden naar rechts.

a) Neem onderstaande binaire boom over en voeg passende draden toe.



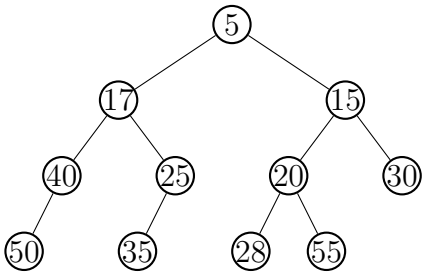
b) Geef een algoritme dat een symmetrische wandeling uitvoert op een bedrade boom, zonder stapel of recursie te gebruiken. Gebruik pseudo-code.
(Neem aan dat de draden permanent zijn, dus *niet* methode Morris.)

c) Neem aan dat de boom een binaire zoekboom is. Laat zien hoe we een (willekeurige) waarde aan de boom wordt toegevoegd.
Zorg ervoor dat daarbij de bedrading van de boom correct blijft. (Een schetsje helpt.)

2. **a)** De operatie *DecreaseKey* in de ADT priority (min-)queue is 'speciaal', in welke zin?

Leftist trees vormen de implementatie van de priority queue, met 'ritsen' als basisoperatie (*zip*).

b) Hieronder is een leftist tree gegeven voor een min-heap.
Welke leftist tree ontstaat uit deze boom als we **(i)** element 10 toevoegen, en als we **(ii)** het minimum verwijderen? (weer in de oorspronkelijke boom)



c) Beredeneer dat de standaard queue operaties *DeleteMin* en *Insert* in logaritmische tijd werken. Wees precies: welke eigenschap van leftist trees gebruik je?

3. **a)** Wat is een topologische ordening op de knopen in een gerichte graaf?

b) Hieronder staat een schematische recursieve functie voor *depth-first search* van grafen, aan te roepen zolang er knopen zijn die nog niet eerder bezocht werden.

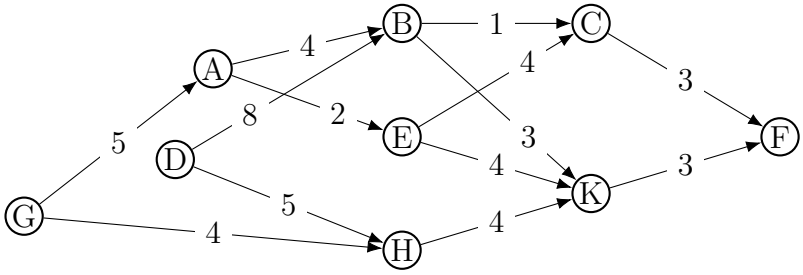
depth-first

```
DFS (KnoopType x)
    // aanname: knoop x nog niet bezocht
    bezoek x
    markeer x als bezocht
    for elke knoop w bereikbaar vanuit x
    do  if (w niet bezocht)
        then DFS (w)
        fi
    od
    // knoop x volledig afgehandeld
end // DFS
```

Geef aan hoe deze functie gebruikt kan worden bij het bepalen van een topologische ordening van een acyclische graaf.

In onderstaande graaf representeren de takken projecten. De benodigde tijdsduur is als label gegeven. Een project (i, j) kan pas worden uitgevoerd als alle projecten eindigend in knoop i zijn afgelopen, voor het overige kunnen projecten parallel uitgevoerd worden.

c) Bepaal, met behulp van een topologische ordening van de graaf, het vroegste tijdstip waarop alle projecten afgerond kunnen zijn, als we beginnen op tijdstip 0. Laat duidelijk zien hoe je aan je (tussen-)resultaten komt.



4. 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.

a) Wat is de tijdscomplexiteit (O) van de volgende operaties?
(i) Naïef zoeken naar een patroon in een tekst
(ii) Knuth-Moris-Pratt (KMP) opstellen links en patroon zoeken.

b) Geef een efficient algoritme dat de *failure-links* voor een patroon bepaalt.

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

d) We zoeken naar P in de tekst T = `BBAA BBAB BABB BBAB BBAB` (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?