Gevraagde functies en programma's mogen in pseudo-code gegeven worden.
Geef steeds voldoende uitleg.

1. **a)** Beschrijf de abstracte datastructuur *Priority Queue*: wat wordt opgeslagen, wat zijn de standaard operaties (en wat doen ze)? Abstract!

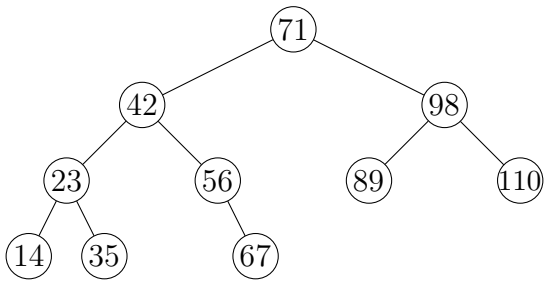
b) Beschrijf de implementatie *Leftist tree* voor Priority Queues. Wat is de structuur van de bomen? Beschrijf de basisoperatie (*ritsen*). Hoe worden verwijderen en toevoegen met deze basisoperatie uitgevoerd?

c) Iets geheel anders. Als we bij een binaire zoekboom op efficiënte wijze de k-de sleutel willen vinden hoe kan de boom dan eenvoudig worden aangepast?

2. Deze opgave gaat over AVL-bomen.

a) Waarom zijn AVL-bomen te prefereren boven standaard binaire zoekbomen?

We gaan sleutels toevoegen en verwijderen. Beschouw de volgende boom B.



b) i. Wat is het resultaat van het toevoegen van sleutel 7 aan boom B?

ii. Wat is het resultaat van het toevoegen van achtereenvolgens 70 en 69 aan (de oorspronkelijke) boom B?

c) i. We willen de wortel 71 uit boom B verwijderen. Hoe doen we dat? Er zijn twee (symmetrische) manieren.

ii. Wat is het resultaat van het resultaat van het verwijderen van de twee grootste sleutels 110 en 98 uit (de oorspronkelijke) boom B?

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

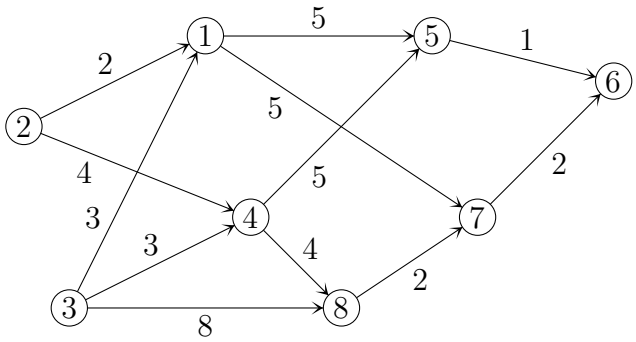
b) Hieronder staat een schematische recursieve functie voor *depth-first search* van grafen, aan te roepen zolang er knopen zijn die nog niet eerder bezocht werden.

```
DFS (KnoopType x)
{   // aanname: knoop x nog niet bezocht
    bezoek x
    markeer x als bezocht
    for (elke knoop w bereikbaar vanuit x)
    {    if (w niet bezocht)
         {   DFS (w)
         }
    }
    // knoop x volledig afgehandeld
}
```

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

c) In onderstaande graaf representeren de takken projecten. 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. De gewichten op de takken zijn de benodigde tijdsduren.

Bereken, 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. **a)** Welk probleem lost het algoritme van Huffman op?
Als er n sleutels gegeven zijn, wat is dan de complexiteit? Geef enige toelichting.

b) Pas Huffman toe op symbolen $a, b, \ldots, f$ met respectievelijke frequenties $10, 7, 2, 8, 11, 5$.

c) Er bestaat een *dynamische* versie van Huffmans algoritme, waar de frequenties kunnen veranderen. Daartoe wordt gebruik gemaakt van een bijzondere eigenschap van de bomen die geconstrueerd worden: de waardes in de knopen kunnen *breadth-first* van groot naar klein gekozen worden. Leg uit dat dit inderdaad lukt.