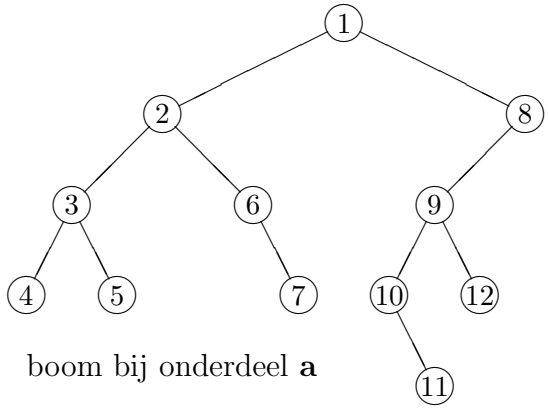
Gevraagde functies en programma's mogen in pseudo-code gegeven worden.
Geef steeds voldoende uitleg.

1) **a.** Neem onderstaande binaire boom over en breng er de symmetrische bedrading (alleen rechterdraden) in aan.

```
struct Knoop
{    int       Info;
     Knoop     *Links, *Rechts;
     TagType   RTag;      // alleen bij b.
     int       Prenum;    // alleen bij c.
     int       Niveau;    // alleen bij d.
};
```



boom bij onderdeel **a**

De knopen in een symmetrisch bedrade binaire boom zijn van de gegeven structuur. Hierbij is `TagType` een enumeratie-type dat de waardes `Tak` en `Draad` kent. We gaan onderzoeken met welke alternatieve velden we deze bomen kunnen representeren.

b. Het veld `RTag` geeft aan of de pointer `Rechts` tak dan wel draad is. De wortel van de boom is van het type `Knoop*` en heet `Wortel`. In principe kan de boom leeg zijn.
Geef een algoritme om de Info-velden van de knopen in de boom in de symmetrische (=LWR) volgorde af te drukken. Gebruik de draden, geen recursie of stapel.

c. Veronderstel nu dat iedere knoop in een symmetrisch bedrade boom een veld `Prenum` heeft, dat het nummer van de knoop in de pre-orde (=WLR) volgorde bevat.
Laat zien dat we het veld `RTag` nu niet nodig hebben om te bepalen of een pointer een tak of een draad is.

d. Veronderstel nu dat iedere knoop in een symmetrisch bedrade boom een veld `Niveau` heeft, dat het niveau van de knoop in de boom bevat (de wortel zit op niveau 1, zijn kinderen op niveau 2, enzovoort).
Laat zien dat we het veld `RTag` ook in dit geval niet nodig hebben om te bepalen of een pointer een tak of een draad is.

e. Stel je moet een dynamische boom beheren waar bladeren zowel kunnen worden verwijderd als toegevoegd. Welk veld `RTag`, `Prenum`, of `Niveau` is duidelijk ongeschikt? Vergelijk ook de overige twee keuzes.

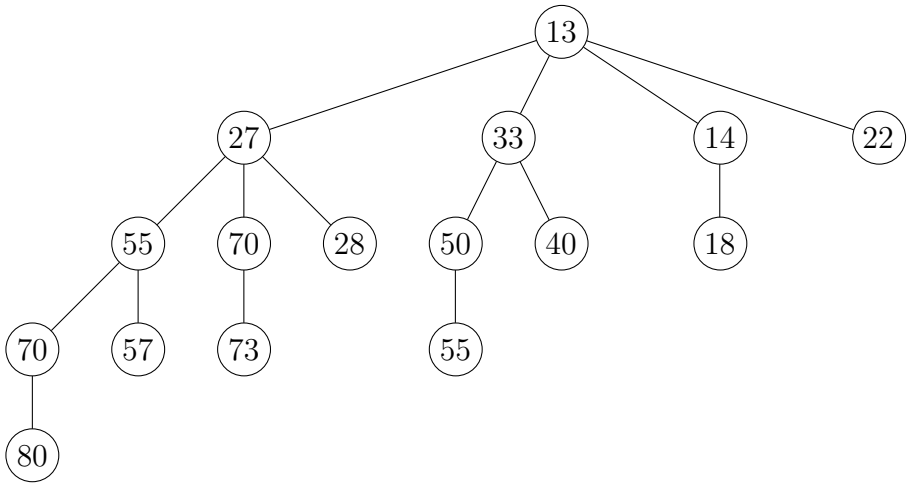
2) **a.** Geef de definitie van priority queue. Vermeld de operaties inclusief parameters, pre- en postcondities en een (informele) beschrijving.

De binomial queue en Fibonacci queue zijn implementaties van dit abstracte datatype.

b. i. Construeer een *binomial queue* door, uitgaande van een lege queue, achtereenvolgens de volgende getallen toe te voegen: 17, 33, 41, 3, 7, 6, 53, 9 en 11. Uiteraard dient ook het tussenresultaat na iedere toevoeging een binomial queue te zijn. Geef via tussenresultaten en een korte toelichting duidelijk aan hoe je aan je antwoord komt.

ii. Voer op het resultaat van het vorige onderdeel twee maal de operatie DeleteMin (VerwijderMin) uit.
Geef ook hier via tussenresultaten en een korte toelichting duidelijk aan hoe je aan je antwoord komt.

c. Gegeven is de volgende *Fibonacci queue* Q:

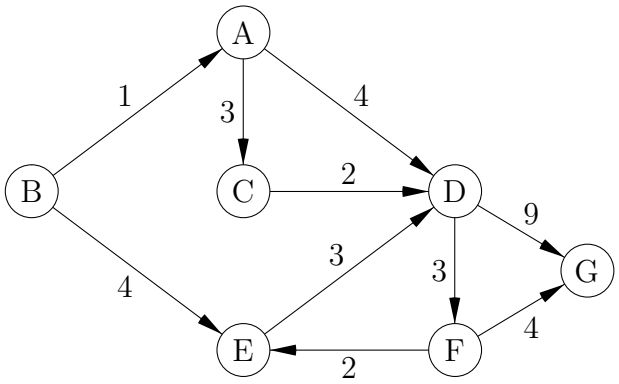


Voer achtereenvolgens de volgende operaties uit op Q: DecreaseKey(28,17), DecreaseKey(57,9), DecreaseKey(70,30), Deletemin() en Insert(23). Geef ook nu voldoende tussenstappen weer.

nb. Geen van de knopen in Q zijn gemarkeerd; dus geen van de knopen hebben reeds een subboom verloren. DecreaseKey(x,y) betekent: "Sleutelwaarde x wordt verlaagd tot y."

3) **a.** Een (ongerichte) graaf kan in het algemeen meer dan één minimale opspannende boom hebben. Geef aan waar een ongerichte graaf aan moet voldoen om precies één minimale opspannende boom te hebben. (Een bewijs is niet nodig.)

Gegeven is de volgende gerichte graaf G_1:



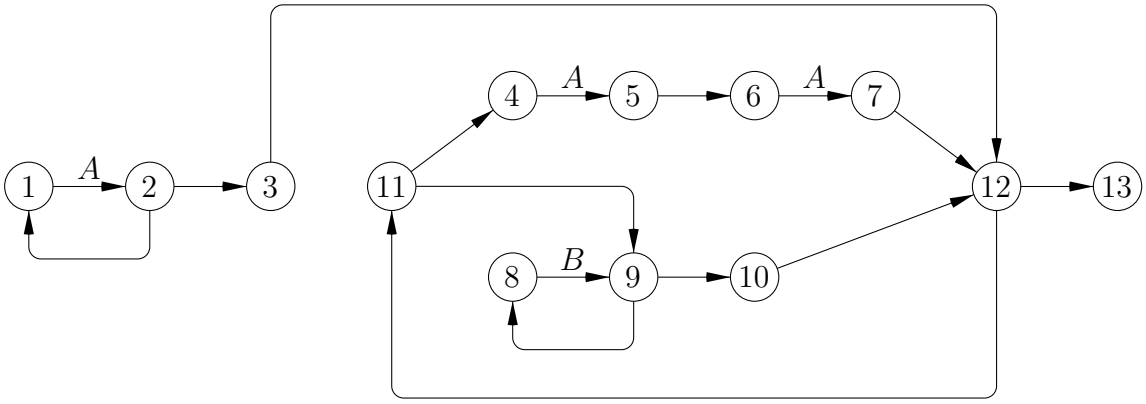
b. Bereken met behulp van Dijkstra's algoritme de kortste paden vanuit knoop A. Geef voldoende tussenstappen weer.

c. Voer het algoritme van topologisch sorteren uit op G_1 en geef aan waarom daar uit blijkt of G_1 al dan niet topologisch sorteerbaar is. Geef ook nu voldoende tussenstappen weer.

d. Geef aan, gegeven een gerichte graaf G, hoe met behulp van het algoritme van Floyd (of een eenvoudige variant daarvan) te bepalen is of er een pad is tussen elk paar knopen i en j in G.

4) **a.** Geef een boom voor de structuur van de reguliere expressies `A*(AA+B*)*` en `A*B+BAB+B*AA`.

Onderstaande automaat werd geconstrueerd voor `A*(AA+B*)*`.
Niet aangegeven zijn de begintoestand 2 en de eindtoestand 13.



b. Belangrijke toestanden zijn die net ná een letter, de begin- en de eindtoestand. Bepaal voor elk van de toestanden uit de verzameling $\{2, 5, 7, 9, 13\}$ en elk van de letters A, B welke toestanden uit die verzameling bereikt kunnen worden via die letter. Voer hiervoor steeds een wandeling in de graaf uit vanuit een gekozen toestand op zoek naar letters.

Plaats de gevonden gegevens in een matrix. Hieronder een voorbeeld; de gegeven waarden moeten nog wel gereconstrueerd worden.

	2	5	7	9	13
A	$2, 5$	...	...		
B	9	...	...		

c. Na het vorige onderdeel hebben we de automaat zelf niet meer nodig om te zoeken of een string aan de taal voldoet. De matrix 'van-letter-naar' is voldoende om te kunnen zoeken.

Ga met behulp van de matrix na of de strings `AABA` en `ABAABBAA` aan de expressie `A*(AA+B*)*` voldoen.