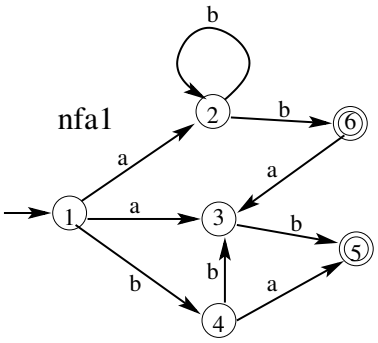
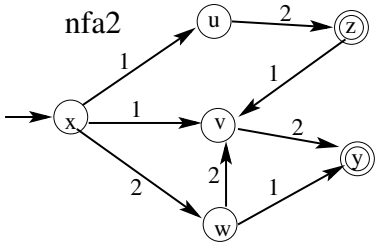
Haskell gekwoteerde oefening
22 mei 2013

KULAK

1 De $\epsilon-$vrije NFA

Een $\epsilon-$vrije niet-deterministische eindige toestandsautomaat kan voorgesteld worden door een gerichte graaf met namen voor de knopen - die ook toestanden worden genoemd - en op elke boog een teken. Er is één knoop aangeduid als startknoop (in de tekening de knoop waar een boog uit het niets aankomt) en alle knopen hebben de aanduiding of ze accepterend zijn of niet (in de tekening is *accepterend* aangeduid met een dubbele cirkel).
Een sequentie van tekens wordt aanvaard door een NFA als en slechts als er een pad bestaat van de beginknoop naar een accepterende knoop zodat de tekens op de gevolgde bogen in dat pad juist die sequentie opleveren. Hieronder twee zulke NFA's.





Kijk eerst na of je de volgende uitspraken begrijpt ivm nfa1 en inziet dat ze correct zijn:

- 6 is een accepterende toestand en 3 niet
- de sequentie abbbb wordt aanvaard en baaa niet
- de sequentie ab wordt op twee *manieren* aanvaard
- elke sequentie die door de gegeven NFA wordt aanvaard is eindig, maar er is geen bovengrens op de lengte van de sequenties die aanvaard worden

Nu ben je klaar voor de opgave ...

2 De opgave

Je zal een aantal voorbeeld $\epsilon-$vrije NFA's krijgen in de vorm van een Haskell functie die $nfai$ (i een getal) heet en geen argumenten heeft. Voor de figuur hiervoor is dat

```
nfa1 = NFA 1 [5,6] [Boog 1 'a' 2, Boog 2 'b' 2, Boog 2 'b' 6,
                    Boog 6 'a' 3, Boog 3 'b' 5, Boog 1 'a' 3,
                    Boog 1 'b' 4, Boog 4 'b' 3, Boog 4 'a' 5]

nfa2 = NFA 'x' ['y','z'] [Boog 'x' 1 'u',                Boog 'u' 2 'z',
                          Boog 'z' 1 'v', Boog 'v' 2 'y', Boog 'x' 1 'v',
                          Boog 'x' 2 'w', Boog 'w' 2 'v', Boog 'w' 1 'y']
```

waarbij de volgorde in de lijsten van geen belang is.
[nfa2 is gelijk aan nfa1 behalve dat de types van de tekens en knopen anders zijn, en dat de boog van 2 naar 2 uit nfa1 verwijderd is]
Er wordt gevraagd te schrijven:

1. een functie *accepteert* die gegeven een sequentie (een lijst) van tekens en een NFA aangeeft (True of False) of die sequentie aanvaard wordt

   ```
   *Main> accepteert nfa1 ``abbbb''
   True
   *Main> accepteert nfa1 ``baaa''
   False
   ```

2. een functie *meermaals* die die gegeven een sequentie van tekens en een NFA aangeeft (True of False) of die sequentie aanvaard wordt op meer dan één manier

   ```
   *Main> meermaals nfa1 ``abbbb''
   False
   *Main> meermaals nfa1 ``baaa''
   False
   *Main> meermaals nfa1 ``ab''
   True
   ```

3. een functie *oneindig* die gegeven een NFA aangeeft (True of False) of de NFA sequenties van willekeurig grote lengte kan aanvaarden: oneindig nfa1 moet True geven

   ```
   *Main> oneindig nfa1
   True
   *Main> oneindig nfa2
   False
   ```

4. de meest algemene typering van alle gevraagde functies en de gebruikte types, zodanig dat ook andere types van knoopnamen en tekens kunnen gegeven worden

5. een functie *toplevel* die als argument een NFA krijgt en

   - een string (dus een rij characters) inleest
   - uitschrijft "geaccepteerd" indien de gegeven NFA de string accepteert, en anders "verworpen"; hieronder staan de lijnen met user input aangegeven met een leidende i (die niet bij de input hoort)

```
        *Main> toplevel nfa1
i       ab
        geaccepteerd
        *Main> toplevel nfa1
i       abbbbb
        geaccepteerd
        *Main> toplevel nfa1
i       baaaa
        verworpen
        *Main> toplevel nfa1
i       ba
        geaccepteerd
```

   je geeft natuurlijk ook de meest algemene typering van die functie

Als je een hint denkt te kunnen gebruiken, dan moet je daarom publiek vragen. Veel plezier !