```
-- Testfaelle zu Aufgabenblatt 7


-- Testfaelle 1a: Eq BTree

(BLeaf 1 "a") == (BLeaf 2 "a")
(BLeaf 1 "a") /= (BLeaf 1 "ab")
(BLeaf 1 "a") /= (BNode 3 "a" (BLeaf 1 "a") (BLeaf 2 "ab"))
(BNode 3 "a" (BLeaf 1 "a") (BLeaf 2 "ab")) == (BNode 4 "a" (BLeaf 5 "a") (BLeaf 6
"ab"))


-- Testfaelle 1b: Eq LTree


(LNode 0 2 [] == LNode 3 2 [])
(LNode 11 1 [(LNode (x+5) x) [] | x <- [1..10]]) == (LNode 11 1 [(LNode (x+5) x)
[] | x <- [1..10]])
(LNode 11 1 [(LNode (x+5) x) [] | x <- [1..10]]) /= (LNode 11 1 [(LNode (x+33) x
[(LNode ((100*x)+y-30) (y+2) []) | y <- [0..x]] ) | x <- [1..10]])
(LNode 11 1 [(LNode (x+33) x [(LNode ((100*x)+y-30) (y+2) []) | y <- [0..x]] ) | x
<- [1..10]]) == (LNode 11 1 [(LNode (x+5)  x [(LNode ((100*x)+y+10) (y+2) []) | y
<- [0..x]] ) | x <- [1..10]])


-- Testfaelle 2a: instance Structure BTree

(noOfSources (BNode 1 "a" (BLeaf 2 "b") (BLeaf 3 "c"))) == 1
(noOfSinks (BNode 1 "a" (BLeaf 2 "b") (BLeaf 3 "c"))) == 2
(notSourceConnected (BNode 1 "a" (BLeaf 2 "b") (BLeaf 3 "c"))) == []
(notSinkConnected (BNode 1 "a" (BLeaf 2 "b") (BLeaf 3 "c"))) == []


-- Testfaelle 2b: instance Structure LTree

(noOfSources (LNode 1 "a" [LNode 2 "b" [],LNode 3 "c" []])) == 1
(noOfSinks (LNode 1 "a" [LNode 2 "b" [],LNode 3 "c" []])) == 2
(notSourceConnected (LNode 1 "a" [LNode 2 "b" [],LNode 3 "c" []])) == []
(notSinkConnected (LNode 1 "a" [LNode 2 "b" [],LNode 3 "c" []])) == []



-- Testfaelle 2c: instance Structure ALgraph

(noOfSources (ALg [(1,[2]),(2,[1]),(3,[2])])) == 1
(noOfSinks (ALg [(1,[2]),(2,[1]),(3,[2])])) == 0
(notSourceConnected (ALg [(1,[2]),(2,[1]),(3,[2])])) == []
(notSinkConnected (ALg [(1,[2]),(2,[1]),(3,[2])])) == [1,2,3]



-- Testfaelle 3:

-- alle wörter über {a,b,c,d,e}*, die weder bc noch cb enthalten, und nicht mit
einem "a" enden

-- zustand 0 "keine Gefahr"
-- zustand 1 "a ist am Ende"
-- zustand 2 "Gefahr, cb zu generieren"
-- zustand 3 "Gefahr, bc zu generieren"
```

```
-- (AMg [(["de","a","c","b"]),(["de","a","c","b"]),(["de","a","c",""]),
(["de","a","","b"])])
-- start: 0
-- end:   0,2,3

( accept (AMg [(["de","a","c","b"]),(["de","a","c","b"]),(["de","a","c",""]),
(["de","a","","b"])]) 0 [0,2,3] "abbbacccdededeebab" ) == True
( accept (AMg [(["de","a","c","b"]),(["de","a","c","b"]),(["de","a","c",""]),
(["de","a","","b"])]) 0 [0,2,3] "abbbacccdededeecbab" ) == False
( accept (AMg [(["de","a","c","b"]),(["de","a","c","b"]),(["de","a","c",""]),
(["de","a","","b"])]) 0 [0,2,3] "abbbacccdededeebaba" ) == False
( accept (AMg [(["de","a","c","b"]),(["de","a","c","b"]),(["de","a","c",""]),
(["de","a","","b"])]) 0 [0,2,3] "abbbacccdededeebcab" ) == False
( accept (AMg [(["de","a","c","b"]),(["de","a","c","b"]),(["de","a","c",""]),
(["de","a","","b"])]) 0 [0,2,3] "" ) == True
```