Aufgabe 1)
Funktion 1 : es handelt sich um eine Funktion, da (a,b) ein Tupel ist und somit nicht vertauscht werden kann.(z. B. C-a = b)
Funktion 2 : es handelt sich um keine Funktion, da es sich um eine gleichung zwischen den tupeln (a,b,c) und (d,e,f) handelt
Funktion 3 : es handelt sich um keine funktion, da es sich um eine menge von quadratischen funktionen handelt
Funktion 4 : es handelt sich um keine Funktion, da keinem Wert ein neuer Wert zugewiesen wird
Funktion 5 : es handelt sich um eine Funktion, da b element von S ist(welche eine funktion ist), und a in relation zu b steht.

2.)
1. Der Datentyp einer Variable zeigt an, welchen Wert die variable jetzt oder später im Verlauf des Programms erhalten kann.
Folgende Arten von Datentypen existieren :

Aufzählungstypen : Sind Aufzählungen von Werten als ein Datentyp.
Beispiel sind hier Wochentage. Data Weekday = Mo | Tu | We | Th | Fr | Sa | Su

Produkttypen : In Haskell sind Produkttypen als "Tupeltypen" vordefiniert. Damit sich lassen sich beispielsweise Koordinaten darstellen oder auch ein Datum(als Tupel) darstellen.
Beispiel Koordinaten :
type Point = (Double, Double)
type Complex (Double, Double)
Beispiel Datum :
type Date = (Tag, Monat, Jahr)

Summentypen : Summentypen setzen Wertemengen als disjunkte Vereinigung andere Wertemengen zusammen. Hierbei können mehrere Produkttypen zusammengesetzt werden.
Beispiel :
type Point (Double, Double)
data Shape = Circ Point Double
| Rectangle Point Point
Für jede Art von Figur gibt es dann auch einen eigenen Konstruktor
Circ :: Point -> Double -> Shape
Rectangle :: Point -> Point -> Shape

2.
a)
data Bruch = Bruch {Zaehler :: Integer, Nenner :: Integer}, **Produkttyp**

data Hausaufgabe = Bestanden | Nichtbestanden | Nachbearbeitet, **Aufzählungstyp**

data Wochentag = Montag | Dienstag | Mittwoch | Donnerstag | Freitag | Samstag | Sonntag , **Aufzählungstyp**

data Monat = Januar | Februar | März | April | Mai | Juni | Juli | August | September | Oktober | November | Dezember, **Aufzählungstyp**

data Datum = Datum{tag :: Wochentag, monat :: Monat, jahr :: Integer}, **Produkttyp**

data Uhrzeit = Uhrzeit {stunden :: Integer, minuten :: Integer}, **Produkttyp**

data Kasse = Kasse {kassenID :: Integer, nachname :: String}, **Produkttyp**

data Preis = Preis {euro :: Integer, cent :: Integer} , **Produkttyp**

data Mensaessen = Tagessuppe Sparessen | MensaVitalEssen | Bioessen | Vegetarisch | Aktion, **Aufzählungstyp**

data Wahlessen = Wahlessen{mensaessen :: Mensaessen, preis :: Preis}, **Summentyp**

data Kassenbon = Kassenbon { kasse :: Kasse, datum :: Datum, uhrzeit :. Uhrzeit, essen :: Wahlessen} , **Summentyp**

Aufgabe 3)
1.
a : a ist T, da loesche T -> Tliste hat und deswegen muss es T sein
b : b ist String , da T nach dem T String und danach Integer kommt
c : c ist demnach Integer, da es folgt auf den String b
d : ist String, gleicher grund wie bei b
e : ist Integer, gleicher grund wie bei c
f : ist Tliste, da NichtLeer T und Tliste hat, (T d e) ist hier T und f ist Tliste

g: existiert nicht, da leer
h : ist T, da nichtleer T
i: ist String, da oben T String Integer definiert ist
j : ist T, da nichtleer T
k : k ist String, da oben T String Integer definiert ist
l : ist Tliste

m : ist Integer, da Integer->Tliste ->VielleichtT und m wird mit 0 verglichen
n : n ist T, da nichtleer T Tliste
o : ist Tliste, da nichtleer T Tliste

2.
```
*Main> loesche (T "Ernie" 7) liste1
NichtLeer (T "Bert" 9) (NichtLeer (T "Ernie" 7) Leer)
```
wurde gelöscht, da String ernie entsprach

```
*Main> ersetze (T "Bibo" 9) liste2
NichtLeer (T "Ernie" 8) (NichtLeer (T "Bibo" 9) Leer)
das 2. wurde ersetzt, da der string "bibo" entsprach

*Main> ersetze (T "Bibo" 9) liste3
NichtLeer (T "Bibo" 9) (NichtLeer (T "Bibo" 9) Leer)
beide wurden ersetzt, da bibo beiden strings entsprach

*Main> findeAnIdx 1 liste3
Doch (T "Bibo" 7)
da m == 0
```