```haskell
-- Marvin Glaser
-- 4424114
-- Gruppe 3
-- Martin Parnet


-- ===============================================================================
-- ==== Aufgabe 1 a)

-- generated all possible tuples with list comprehension and given restrictions
route_faith = [element | element <- [(7, b, c, d, 2, 3, 1) | b <- [4..6],
    c <- [4..6], d <- [4..6], b /= c, b /= d, c /= d, b /= 5, d /= 5, if c == 5
    then d == 4 else True]]

-- Test:
{-
*Main> route_faith
[(7,6,5,4,2,3,1)]
-}
-- there is only one possible route faith can take with the given restrictions


-- ===============================================================================
-- ==== Aufgabe 1 b)

-- copied from worksheet
data Tragetasche = Tragetasche [Lieferung]
    deriving (Eq, Show)
data Lieferung = Lieferung Objekt Bezahlung Zielort
    deriving (Eq,Show)
data Objekt = Objekt Zerbrechlich Gewicht Abmessungen
    deriving (Eq,Show)
type Bezahlung = Float
type Zielort = String
type Zerbrechlich = Bool
type Gewicht = Float
type Abmessungen = (Float,Float,Float)


-- b1)

-- defined given variables as "tasche"
tasche =
    (Tragetasche [Lieferung (Objekt True 30 (50, 50, 50)) 100 "Eden Village",
    Lieferung (Objekt False 15 (299, 324, 5)) 100 "Eden Village"])

-- Test:
{-
*Main> tasche
Tragetasche
[Lieferung (Objekt True 30.0 (50.0,50.0,50.0)) 100.0 "Eden Viallage",
Lieferung (Objekt False 15.0 (299.0,324.0,5.0)) 100.0 "Eden Village"]
-}

-- b2)

-- used pattern matching to isolate weight of package and add up for result
gesamtGewicht :: Tragetasche -> Float
gesamtGewicht (Tragetasche []) = 0.0
gesamtGewicht (Tragetasche (Lieferung (Objekt _ x _) _ _ : [])) = x
gesamtGewicht (Tragetasche (Lieferung (Objekt _ x _) _ _ : xs)) = x
    + gesamtGewicht (Tragetasche xs)

-- Test:
{-
*Main> gesamtGewicht tasche
45.0
-}
```

```haskell
-- b3)
{-
aendereZerbrechlich :: Tragetasche -> Abmessungen -> Tragetasche
aendereZerbrechlich (Tragetasche []) = Tragetasche []
aendereZerbrechlich (Tragetasche (Lieferung (Objekt _ a Abmessungen) b c)) =
    Tragetasche [Lieferung (Objekt True a Abmessungen) b c]
aendereZerbrechlich (Tragetasche [x]) = Tragetasche [x]
-}



-- ===============================================================================
-- ==== Aufgabe 2 a)

-- defined list comprehension with given restrictions
list1 = [x | x <- [1..], (mod x 2 /= 0), (mod x 13 == 3), (mod x 20 == 5)]

-- Tests:
{-
*Main> take 20 list1
[185,445,705,965,1225,1485,1745,2005,2265,2525,2785,3045,3305,3565,3825,4085
4345,4605,4865,5125]

*Main> take 200 list1
[185,445,705,965,1225,1485,1745,2005,2265,2525,2785,3045,3305,3565,3825,4085,
4345,4605,4865,5125,5385,5645,5905,6165,6425,6685,6945,7205,7465,7725,7985,
8245,8505,8765,9025,9285,9545,9805,10065,10325,10585,10845,11105,11365,11625,
11885,12145,12405,12665,12925,13185,13445,13705,13965,14225,14485,14745,15005,
15265,15525,15785,16045,16305,16565,16825,17085,17345,17605,17865,18125,18385,
18645,18905,19165,19425,19685,19945,20205,20465,20725,20985,21245,21505,21765,
22025,22285,22545,22805,23065,23325,23585,23845,24105,24365,24625,24885,25145,
25405,25665,25925,26185,26445,26705,26965,27225,27485,27745,28005,28265,28525,
28785,29045,29305,29565,29825,30085,30345,30605,30865,31125,31385,31645,31905,
32165,32425,32685,32945,33205,33465,33725,33985,34245,34505,34765,35025,35285,
35545,35805,36065,36325,36585,36845,37105,37365,37625,37885,38145,38405,38665,
38925,39185,39445,39705,39965,40225,40485,40745,41005,41265,41525,41785,42045,
42305,42565,42825,43085,43345,43605,43865,44125,44385,44645,44905,45165,45425,
45685,45945,46205,46465,46725,46985,47245,47505,47765,48025,48285,48545,48805,
49065,49325,49585,49845,50105,50365,50625,50885,51145,51405,51665,51925]
-}


-- ===============================================================================
-- ==== Aufgabe 2 b)

-- defined help function to convert tuple to list
-- drew tuples form input, converted tuple to list and then used comprehension
list2 xs = [ x | s <- xs, x <- caster_help s ]

caster_help (x, y) = [x, y]

-- Tests:
{-
*Main> list2 [(1,2),(3,4),(5,6)]
[1,2,3,4,5,6]
*Main> list2 []
[]
*Main> list2 [(0,0),(3,4),(5,6), (8888888,88888888)]
[0,0,3,4,5,6,8888888,88888888]
-}


-- ===============================================================================
-- ==== Aufgabe 2 c)

-- used list comprehension and ordered tuple elements in correct order
-- to get desired incrementation of variables
list3 = [ (a, b, c) | b <- [True, False], a <- [0, 3, 20, 25, 37, 97],
    c <- "abcd"]
```

```haskell
-- Tests:
{-
*Main> list3
[(0,True,'a'),(0,True,'b'),(0,True,'c'),(0,True,'d'),(3,True,'a'),(3,True,'b'),
(3,True,'c'),(3,True,'d'),(20,True,'a'),(20,True,'b'),(20,True,'c'),
(20,True,'d'),(25,True,'a'),(25,True,'b'),(25,True,'c'),(25,True,'d'),
(37,True,'a'),(37,True,'b'),(37,True,'c'),(37,True,'d'),(97,True,'a'),
(97,True,'b'),(97,True,'c'),(97,True,'d'),(0,False,'a'),(0,False,'b'),
(0,False,'c'),(0,False,'d'),(3,False,'a'),(3,False,'b'),(3,False,'c'),
(3,False,'d'),(20,False,'a'),(20,False,'b'),(20,False,'c'),(20,False,'d'),
(25,False,'a'),(25,False,'b'),(25,False,'c'),(25,False,'d'),(37,False,'a'),
(37,False,'b'),(37,False,'c'),(37,False,'d'),(97,False,'a'),(97,False,'b'),
(97,False,'c'),(97,False,'d')]
-}


-- ============================================================================
-- ==== Aufgabe 2 d)

-- generated two seperate lists with the desired elements and united thsoe lists
list4 = [ (a, b, c) | a <- [1..4], b <- [1..4], c <- [(\a b -> a + b) a b],
    even c] ++ [ (a, b, c) | a <- [1..4], b <- [1..4],
    c <- [(\a b -> a - b) a b], even c]

-- Tests:
{-
[(1,1,2),(1,3,4),(2,2,4),(2,4,6),(3,1,4),(3,3,6),(4,2,6),(4,4,8),(1,1,0),
(1,3,-2),(2,2,0),(2,4,-2),(3,1,2),(3,3,0),(4,2,2),(4,4,0)]
-}


-- ============================================================================
-- ==== Aufgabe 2 e)

-- used additional varialbe that incrementally caps the variables in output
-- tuple. Sadly VERY inefficient
list5 = [ (a, b, c, d) | x <- [0..], a <- [0..x], b <- [0..x], c <- [0..x],
    d <- [0..x]]

-- Test
{-
*Main> elem (1, 10, 3, 5) list5
True
*Main> elem (1, 1, 1, 1) list5
True
*Main> elem (0, 0, 0, 0) list5
True
*Main> elem (20, 20, 20, 20) list5
True
-}


-- ============================================================================
-- ==== Aufgabe 3 a)

{-
s := \y -> if (x y) then ((\w -> w) y) else ((\z -> z) z)

  FV(\y -> if (x y) then ((\w -> w) y) else ((\z -> z) z))
= (FV(if (x y) then ((\w -> w) y) else ((\z -> z) z))) \ {y}
= (FV(x y) u FV((\w -> w) y) u FV((\z -> z) z)) \ {y}
= (FV(x) u FV(y) u (FV(\w -> w) u FV(y)) u (FV(\z -> z) u FV(z))) \ {y}
= ({y} u {x} u ((FV(w) \ {w}) u {y}) u ((FV(z) \ {z}) u {z})) \ {y}
= ({y} u {x} u (({w} \ {w}) u {y}) u (({z} \ {z}) u {z})) \ {y}
= ({y} u {x} u ({} u {y}) u ({} u {z})) \ {y}
= ({y} u {x} u {z}) \ {y}
= {z, x}
```

```
  GV(\y -> if (x y) then ((\w -> w) y) else ((\z -> z) z))
= ({y} u (GV(if (x y) then ((\w -> w) y) else ((\z -> z) z))))
= ({y} u (GV(x y) u GV((\w -> w) y) u GV((\z -> z) z)))
= ({y} u (GV(x) u GV(y)) u GV(\w -> w) u GV(y) u GV(\z -> z) u GV(z))
= ({y} u ({} u {y}) u GV(w) u {w} u GV(z) u {z} u {})
= ({y} u ({y}) u {} u {w} u {} u {z})
= {y, w, z}
= {y, w, z}
-}
```