1)

LET T = \ x y . x

LET F = \ x y . y x

a mel jsem nadefinovat Not, disjunkci a overit funkcnost na not(or T F)

2)

definovat v haskellu datovy typ pro Booleovske vyrazy: konstantni hodnoty,volne promenne,negaci,konjunkci,disjunkci. zadefinovat to jako instanci tridy Show (+ jsem pochopil jeste vytvorit funkci,ktera bude volana misto show a zobrazi dany vyraz textove),zaregistrovat jako instanci tridy Eq (definovat operator rovnosti).udelat funkci nneg,ktera pro zadany booleovsky vyraz zneguje jen volne promenne a konstantni hodnoty a napsat jeji typ

3)provest dukaz pro map f (xs ++ ys) = map f xs ++ map f ys

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**řešení:**  
1)

LET Not = \ a . a F (\ v. T)

LET OR = \ a b . a T (\ v. b)

Not(OR T F) = Not( (\a b. a T (\v. b)) T F) = Not(T T (\v. F)) = Not ((\ a b. a) T (\v. F)) = Not T = (\ a. a F (\v. T) T = T F (\v. T) = (\ a b. a) F (\v. T) = F

3)

[] ++ xs = xs (1)

x:xs ++ ys = x:(xs ++ ys) (2)

map f [] = [] (3)

map f (x:xs) = f x : map f xs (4)

a)dokazeme,ze plati pro xs = []

map f ([] ++ ys) = map f ys (1)

map f [] ++ map f ys = [] ++ map f ys (3)

= map f ys (1)

b)predpokladame,ze plati: map f (xs ++ ys) = map f xs ++ map f ys (IH)

c)dokazeme,ze plati pro (x:xs)

map f ((x:xs) ++ ys) = map f (x:(xs ++ ys)) (2)

= f x : map f (xs ++ ys) (4)

= f x : (map f xs ++ map f ys) (IH)

map f (x:xs) ++ map f ys = (f x : map f xs) ++ map f ys (4)

f x : (map f xs ++ map f ys) (2)

Q.E.D.