број поена

1. Написати функцију која броји колико елемената листе је веће од првог елемента дате листе. За празну листу вратити нулу.

   (a) рекурзивно

   (b) користећи функцију foldl

   (c) без рекурзије и било каквог fold-а

   (d) написати QuickCheck тест који проверава да ли прве две имплементације дају исто решење

   countGreater1st :: [a] -> Int
   countGreater1st [1, 2] – 1
   countGreater1st [1, 1] – 0
   countGreater1st [1, 2, 0, 3] – 2

број поена

2. Дат је стринг. Написати функцију (без употребе рекурзије) која проверава да ли постоје два узастопна карактера која су иста.

број поена

3. На почетку је програм у стању чекања. Кад корисник унесе своје корисничко име, програм прелази у стање провере да ли је корисничко име исправно. После провере, програм прелази или у стање где је корисничко име валидно, или у стање грешке где се памти порука о грешци.

   Дефинисати тип података (и ништа више – не имплементирати програм) који тачно дефинише скуп стања у којима овај програм може да се нађе.

   data **Model** = ...

број поена

4. Написати функцију (без употребе рекурзије) која за дату сортирану листу проверава да ли је сортирана у растућем или опадајућем поретку. Ако је сортирана у опадајућем поретку, вратити као резултат GT, ако је сортирана у растућем поретку, вратити LT, ако су сви елементи једнаки, вратити EQ.

   whichSort [] – EQ
   whichSort [1,2,3] – LT
   whichSort [3,2,1] – GT

број поена

5. Написати функцију (без употребе рекурзије) која за дату листу враћа индекс првог елемента те листе који је мањи од датог броја n или Nothing ако такав елемент не постоји.

   firstLessThan :: Int → [Int] → Maybe Int
   firstLessThan 6 [41,16,4,1] – Just 2
   firstLessThan 1 [1,16,4,1] – Nothing