Ci hai provato, ma non ci seri riuscito, non stare a ritentare, se te li passano bene, se no lascia stare

Programming languages – Haskell

Laboratory instruction B (2021/22)

T. Goluch

Exercises should be carried out in the given order. After completing each exercise, please present lecturer result and save the function/program code in the solution file. If any of exercises seems too difficult, you can skip them and move on to the next one. Unless stated otherwise as a solution, include the implemented code to the final solution file.

1) Exercise (0.5 pts.)

Run REPL loop: `ghci -flocal-ghci-history`[1] (Haskell Platform) or `stack ghci`/`cabal repl` (The Haskell Tool Stack) alternatively use the site: https://repl.it/repls/ExtraneousAdorableLink.

Implement `revL` function that reverse elements of the list given as a parameter. For example: `revL` = [3,5,7,9] returns: [9,7,5,3].

2) Exercise (0.5 pts.)

Implement a program in revl.hs file containing function `revL`. Load revl.hs file during running ghci and call function: `revL [3,5,7,9]` should return result: [9,7,5,3].

3) Exercise (0.5 pts.)

Add `main` function to revl.hs file printing result of call of function: `revL [2 .. 999]` and then compile using ghc and run executable file.

4) Exercise (0.5 pts.)

Create constants `last` = 999, `beg` = 2, and list `numL` containing increasing integers from 2 to `last`.
Create list `num2L` containing integers from `numL` with first integer and its multiples removed from the list. For example, for the list `numL` = [2,3,4,5,6,7,8,9], in result `numL` = [3,5,7,9], 2 (head - first element) and 4, 6 and 8 will be removed.

5) Exercise (1 pts.)

Implement a function `primes last` returning a list of all prime integers from 2 to `last`. Try to move first element from input list to result list in each step, and then remove all its multiples from the input list. Keep doing this until the list is not empty. For example `primes 20` should return a list:
`[2,3,5,7,11,13,17,19]`

[1] It's temporarily replace the command history file location on the current folder. The file ".ghci_history" can be uploaded as a solution to the exercises implemented in GHCI.

6) Exercise (0.5 pts.)

Implement `triplets` function which takes as a parameter the natural number `n` and generates all $\frac{(2n-1)!}{n!*(n-1)!}$ n-element combinations of the n-element set `n = [1..n]`. For example: `triplets 3` should return: `[(1,1,1),(2,1,1),(2,2,1),(2,2,2),(3,1,1),(3,2,1),(3,2,2),(3,3,1),(3,3,2),(3,3,3)]`

7) Exercise (0.5 pts.)

Complete the function call: `filter … (triplets 5)` specifying the appropriate parameter so as to obtain 3-elements tuple of the right triangle sides lengths from the list of triples generated by the function `triplets` passed as the third parameter.

8) Exercise (0.5 pts.)

Implement `divRem` taking two integers as parameters, which will print a tuple containing the integer division result on the first position and the remainder on the second. For example: `divRem 46 7` should return: `(6,4)`

9) Exercise (0.5 pts.)

Implement `uniq` function returning a list (taken as a parameter) of elements without repetition. For example: `uniq [3,2,6,0,3,6,6,2,1]` should return: `[3,2,6,0,1]` or similar (a different order of elements is allowed).

10) Exercise (0.5 pts.)

Narrow function `uniq` from previous exercise to `Int` types by adding the appropriate header, Checking `> :t` type of the function should be: `uniq :: [Int] -> [Int]`.

11) Exercise (0.5 pts.)

Implement the `setUni` function returning the sum of sets a∪b given as lists `a` and `b`. For example, setUni `setUni [3,5,6] [1,2,6]` should return `[3,5,6,1,2]`. You can use the `uniq` function from the previous task.
Please do not use the `\\` operator, the `union` and `intersect` functions from the Data.List module.

12) Exercise (0.5 pts.)

Implement the `digCount` function printing the number of digits of the number given as the first parameter. For example: `digCount 2354747600` should return: `10`.

13) Exercise (1 pts.)

Implement `permuteL` function printing all (length !) permutations of the list given as a parameter. You can reimplement the previous task and instead of adding the second element to the pair, add all (length - 1) ! permutations of the list with removed element. This will require a recursive call of `permuteL` and handle the situation ending recursion (passing an empty list as a parameter). For example: `permuteL [1,2,3]` should return: `[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]`

14) Exercise (0.5 pts.)

Implement `lexOrd m n` returning the nth number from the lexicographic order of numbers from 01…m to m(m-1)…0 (the number is simply any permutation of digits from 0 to m). For example: `lexOrd 3 2` should return: `0132`