1 Care este signatura unei funcții f care primește ca parametru o listă de șiruri de caractere și calculează lista de lungimi asociate fiecărui șir?

(a) f :: [Char] → Int

(b) f :: String → [Int]

(●) f :: [String] → [Int]

(d) nu se poate defini o astfel de funcție

2 Ce eroare este în codul de mai jos?

```
h x = x + g
  where g x = x + 1
```

(●) funcția g nu este apelată corect

(b) x declarat de mai multe ori

(c) indentarea este greșită

(d) codul este corect

3 Care din următoarele instrucțiuni este o funcție anonimă?

(●) \x . x ^ 2

(b) \x, y → (x + y) ^ 2 \x y -> (x + y) ^ 2 asta ar fi fost sintaxa corecta

(c) f x = x * 3

(d) nicio variantă

4 Care din implementările de mai jos este corectă pentru funcția hof?

hof :: (a → b → c) → (a → b) → a → c

(●) hof h f a = h a (f a)

(b) hof h f a = h a

(c) hof h f a = h f a

(d) nicio variantă

5 Ce valoare are x?

```
l1 = [2, 4..]
l2 = ['a', 'b'..]
l3 = zip l1 l2
x = head.tail l3
```

(a) (4, 20)

(b) [(b, 'b')]

(●) (4, 'b')

(d) operațiile au erori

6 Cu ce operație putem obține valoarea ('b', 4)?

l1 = ['a','b'..]
l2 = [2, 4..]
l3 = take 3 $ zip l1 l2

(a) l1 ++ l2

(b) l3.1

(c) l3[1]

(●) nicio variantă

7 Fie operatorul (<+). Care funcție e o secțiune dreapta pentru acesta?

(<+) :: String → [Int] → Bool

(a) (<+ ['1', '2', '3'])

(b) (<+ "abc)

(●) (<+ [1, 2, 3])

(d) nicio variantă

A right section of an operator allows you to partially apply the operator with its right-hand argument fixed.
For a binary operator like (<+), which has the type signature:
haskell
(<+) :: String -> [Int] -> Bool

A right section would look like this:
haskell
(<+ value)

Where value is the fixed second argument of type [Int].

Fixed second argument: [1, 2, 3] (this is a list of integers, which is the correct type).
Expected second argument type: [Int].
This is a valid right section, as we are fixing the second argument ([1, 2, 3]), and the resulting function takes a String and returns a Bool.

8 Ce tip de date are expresia?

filter(\(x, y) → x == y) [("aa", "aa"), ("b", "bb"), ("abc", "d")]

(●) [([Char], [Char])]

(b) [(Char, Char)]] → [[(Char, Char)]]

(c) [Char] → [(Char, Char)]

(d) [[(Char, Char)]]

9 Ce returnează instrucțiunea?

filter (== "A_") ["Ana", "Are", "Mere"]

(a) ["Mere"]

(●) []

(c) ["Ana", "Are"]

(d) instrucțiune invalidă

filter (\s -> head s == 'A') ["Ana", "Are", "Mere"]

10 Ce tip are expresia?

map (: ["a", "b", "c"])

(a) (a → b) → [a] → [b]

(b) [Char] → [[Char]]

(●) [[Char] → [[[Char]]]

(d) nu se poate evalua

In the expression (: ["a", "b", "c"]), the cons operator is partially applied.
This means we are fixing the second argument (["a", "b", "c"]) as the list, so the function becomes:

(: ["a", "b", "c"]) :: a -> [[Char]]

It takes an element of any type a, and returns a list where that element is added to the front of the list ["a", "b", "c"], r
esulting in a new list of type [[Char]] (a list of strings, or List of List of Char).

11 Care din operații produce rezultatul [10, 11, 12, 13]

(a) map (+1) [10, 11, 12, 13]

(b) map 0 [10, 11, 12, 13]

(●) map id [10, 11, 12, 13] The id function returns its argument unchanged. Therefore, this will produce [10, 11, 12, 13], which is correct.

(d) nicio variantă

12 Ce calculează funcția f?

f xs = foldr (&&) True [x 'mod' 3 > 0 | x ← xs]

(a) definiție incorectă

(●) dacă nu există numere divizibile cu 3 în listă If there are no numbers divisible by 3, all values in the generated list will be True, and thus the result will be True.

(c) dacă exista cel puțin un număr indivizibil cu 3 în listă

(d) nicio variantă

13 Care din următoarele instrucțiuni va întoarce un rezultat?

(a) take 3 $ foldr (^) 2 [1..] foldr doesn't produce a result (due to the infinite list), take 3 cannot operate on this result and therefore won't give anything either

(b) take 3 . foldr (^) 2 [1..] foldl applied to an infinite list will also not terminate, and the result cannot be computed.

(c) take 3 . foldl (^) 2 [1..]

(●) nicio variantă

14 Care sunt constructorii de date în următorul tip de date algebric?

data Arb a = Frunza | Nod a (Arb a) (Arb a)

(a) Frunza, Nod, Arb

(●) Frunza, Nod

(c) A4

(d) Arb

15 Care dintre definițiile de mai jos este corectă?

(a) data Tree = empty | leaf a | branch (Tree a) (Tree a)

(b) data Tree a = Empty | Branch Tree Tree

(c) data Tree = Lead a | Nod b

(●) data Tree a = Empty | Branch a (Tree a) (Tree a)