

**1 Care este semnatura unei funcții f care primește ca parametru o listă de șiruri de caractere și calculează pentru fiecare șir dacă este format numai din vocale sau nu?**

- (a)  $f :: [\text{Char}] \rightarrow \text{String}$
- (b)  $f :: [[\text{Char}]] \rightarrow [\text{Bool}]$
- (c)  $f :: [\text{String}] \rightarrow \text{Bool}$
- (d) nu se poate defini o astfel de funcție

**2 Ce eroare este în codul de mai jos?**

```
f x = x + g x + y
  where g x = x + 1; y = x
```

- (a) codul este corect
- (b) codul este formatat greșit
- (c) x declarat de mai multe ori
- (d) y declarat greșit

**3 Care din cele de mai jos este un exemplu de lazy evaluation?**

- (a) `take 6 [0..]`
- (b) `False && _ = False`
- (c) `take 3 [1, 2, 3, undefined, 4]`
- (d) toate variantele

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

$z :: a \rightarrow (a \rightarrow b) \rightarrow a$

- (a)  $z\ a\ f = f\ a$
- (b)  $z\ f\ a = f$
- (c)  $z\ a = a$
- (d) nicio variantă

**5 Ce valoare are x?**

```
l1 = [2, 4..]
l2 = ['a', 'b'..]
l3 = zip l2 l1
x = (head.reverse.take 5) l3
```

- (a) (2, 'a')
- (b) (10, 'e')
- (c) ('e', 10)
- (d) operațiile au erori

**6 Cum putem obține poziția elementului 'd' din l1?**

`l1 = ['a','b'..]`

- (a) `l1!!3`
- (b) `l1!!'d'`
- (c) `filter(\(a, b) → b == 'd') (zip l1 [0..])`
- (d) nicio variantă

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

`(<>+) :: Bool → [Char] → Int`

- (a) `(<>['a', 'b', 'c'] +)`
- (b) `(<>+ "abc")`
- (c) `(False <>+)`
- (d) `(True <>+ ['x', 'y'])`

**8 Ce tip de date are x?**

`f = filter \(a, b) → b > 'a'`

`second (x, y) = x`

`g = map (second)`

`x = (g.f) [(7, 'a'), (9, 'f'), (24, 'a')]`

- (a) `9`
- (b) `String`
- (c) `Int`
- (d) `[Int]`

**9 Care dintre funcții are semnatura [Int] → Int?**

- (a) `map (+0)`
- (b) `filter (==0)`
- (c) `map (+3).filter (/=7)`
- (d) nicio variantă

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

- (a) `map (-1) [14, 13, 11, 10]`
- (b) `map id [10..13]`
- (c) `map id [13..10]`
- (d) `map(\n → n - 1).reverse.map (+1) $ [10, 11, 12, 13]`

**11 Ce valoare va avea x?**

```
f xs = foldr (||) True [x `mod` 5 > 0 | x <- xs]
x = f [15, 0, 4355, 5, 50]
```

- (a) [False, False, False, True, False]
- (b) True
- (c) False
- (d) eroare de compilare

**12 Care din următoarele instrucțiuni va întoarce un rezultat?**

- (a) foldr (\*) 2 [1..]
- (b) take 3 \$ foldr (\x l → (x + 7):l) [2] [4..]
- (c) take 3 \$ foldl (\l x → (x + 7):l) [2] [4..]
- (d) nicio variantă

**13 Care sunt constructorii de date în următorul tip de date algebric?**

```
data Semaphore a = Red a | Yellow a | Green a
```

- (a) Semaphore
- (b) Green, Yellow, Red
- (c) a
- (d) Semaphore, Red, Yellow, Green

**14 Care dintre variante este un constructor de tip?**

```
data User a = Null | Guest a | Account String a
```

- (a) User
- (b) Account
- (c) a
- (d) nicio variantă

**15 Care dintre definițiile de mai jos este corectă?**

- (a) data MaybeEither a b = Left a | Right c
- (b) data maybeeither = nothing | left | right
- (c) data MaybeEither a b = Nothing | Left a | Right b
- (d) data MaybeEither b = Nothing | Left a | Right b