Test PP Numele ..... Grupa .....

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
a																				
b																				
c																				
d																				

1. Care este valoarea următoarei expresii în Haskell?

```
take 4 $ iterate (\x -> (head x + 1) : x) [1]
```

   (a) `[[1], [2,1], [3,2,1], [4,3,2,1]]`
   (b) `[[1], [2], [3], [4]]`
   (c) `[[1], [1,2], [1,2,3], [1,2,3,4]]`
   (d) `[1,2,3,4]`

2. Care este valoarea următoarei expresii în Racket?

```
(foldl (lambda(x y) (x y (x y))) 2 (list * +))
```

   (a) `8`
   (b) `2`
   (c) `16`
   (d) `6`

3. Care dintre următoarele funcții este echivalentă cu `func` în Racket?

```
(define (func f L)
  (foldl (lambda (x acc) (cons (f x) acc)) '() L))
```

   (a) `(compose reverse map)`
   (b) `(compose reverse apply)`
   (c) `map`
   (d) `apply`

4. Care este valoarea următoarei expresii în Racket?

```
(foldl
 (lambda (f L) (f cons '() L))
 '(1 2 3)
 (list foldl foldr))
```

   (a) `'(3 2 1)`
   (b) `'(1 2 3)`
   (c) `'(2 3 1)`
   (d) `'(3 1 2)`

5. Ce va afișa următorul program Racket?

```
(define x 1)
(define y 2)
(let* ([x 42] [y (+ y 1)] [z (+ x 1)])
  (+ x y z))
```

   (a) `88`
   (b) `47`
   (c) `6`
   (d) Programul intră în buclă infinită.

6. Ce va afișa umătorul program Racket?

```
(define x 2)
(let ([x 1] [y 2] [f (delay (lambda (y) (+ x y)))])
  (let ([x 5]) ((force f) x)))
```

   (a) `7`
   (b) `3`
   (c) `4`
   (d) `5`

7. Ce va afișa umătorul program Racket?

```
(define (f x) (x))
(f (lambda () (delay 42)))
```

   (a) `#<promise>`
   (b) `#<procedure>`
   (c) `42`
   (d) Eroare

8. Fie tipul Haskell:

```
data Point
    = Point Int Int
    | Add Point Point
    | Inv Point
```

   Care este tipul expresiei `Add . Inv`?

   (a) `Point -> Point -> Point`
   (b) `Point -> Point`
   (c) `Int -> Int -> Point`
   (d) Eroare de sinteză de tip

9. Care este tipul următoarei funcții în Haskell?

```
f x y = concat (head x ++ tail y)
```

   (a) `[[[a]]] -> [[a]] -> [a]`
   (b) `[[a]] -> [[a]] -> [a]`
   (c) `[a] -> [a] -> [a]`
   (d) `[[[a]]] -> [[[a]]] -> [a]`

10. Care este tipul următoarei expresii în Haskell?

```
map (+) [1,2,3]
```

   (a) `Num a => [a -> a]`
   (b) `Num a => [a]`
   (c) `Num a => a`
   (d) Eroare de sinteză de tip

11. Care dintre următoarele afirmații este adevarată?

   (a) Haskell utilizează tipare statică cu inferență de tip.
   (b) Haskell utilizează tipare dinamică cu inferență de tip.
   (c) Haskell utilizează tipare dinamică fără inferență de tip.
   (d) Haskell utilizează tipare statică fără inferență de tip.

12. Care este valoarea următoarei expresii în Haskell?

```
id (id 42) == (id id) 42
```

   (a) `True`
   (b) `False`
   (c) Eroare de sinteză de tip
   (d) Eroare pentru că operatorul `(==)` nu este supraîncărcat

13. Fie următoare definiție în Haskell (ignorați prima linie):

```
{-# LANGUAGE FlexibleInstances #-}
instance (Eq a, Num a) => Eq (a -> a) where
    f == g = (f 1) == (g 1)
```

   Câte elemente va avea lista de mai jos?

```
filter (== id) [(+ 0), (/ 2) . (* 2), \_ -> 1]
```

   (a) 3
   (b) 2
   (c) 1
   (d) 0

14. Fie următoarele definiții în Prolog pentru funcția de maxim:

```
max(X, Y, X):- X > Y, !.
max(_, Y, Y).
```

   Ce afișează următoarele două interogări?

```
max(3, 2, 2).
max(2, 3, 2).
```

   (a) `true` și `false`
   (b) `false` și `true`
   (c) `false` și `false`
   (d) `true` și `true`

15. Ce valoare are `C` în urma rulării, în linia de comandă, a următoarei comenzi Prolog?

```
?- B = 7+4, A is B + 3, Z = A mod B, C = Z + 5.
```

(a) `C = 14 mod (7+4)+5.`
(b) `C = Z + 5.`
(c) `C = (7+4)+3 mod (7+4)+5.`
(d) `ERROR: toplevel: Undefined procedure: (mod)/2`

16. Se dă următorul program în Prolog:

```
prog(L, R) :- acc(L, 0, R).
acc([_], A, A) :- !.
acc([H|T], A, R) :- A1 is A + H, acc(T, A1, R).
```

Ce va produce încercarea de a satisface următorul scop?

```
?- prog([1, 2, 3], R).
```

(a) `R = 3.`
(b) `R = [1, 2, 3].`
(c) `R = 6.`
(d) `R = [3, 2, 1].`

17. Bazat pe faptele de mai jos, care sunt soluțiile query-ului `p(X)` în Prolog?

```
foo(2, 3, 1). foo(2, 3, 6).
foo(1, 2, 3). foo(1, 3, 4). foo(1, 2, 5).
p(X) :- bagof(C, (foo(A, B, C), C mod 2 > 0), [X, Y]).
```

(a) `X = 3.`
(b) `X = 1.`
(c) `X = 3; X = 5.`
(d) `X = 1; X = 3.`

18. Câte soluții va avea următorul scop (de câte ori poate fi satisfăcut)?

```
?- L = [(a1, b1), (a1, b2), (a2, b3)],
   setof(B, member((A, B), L), Sol).
```

(a) 2
(b) 0
(c) 1
(d) Eroare

19. Câte soluții întoarce Prolog pentru interogarea următoare și ce valori are variabila `Z`?

```
member(X, [1,2,3]), Y == X, !, Z is X + Y.
```

(a) zero (eșuează)
(b) una: `2`
(c) trei: `2, 4, 6`
(d) nouă: `2, 3, 4, 3, 4, 5, 4, 5, 6`

20. Care din următoarele expresii NU va produce eroare?

```
Racket: (define x (/ (/ 10 2) (let ([z 0]) z)))
Haskell: let x = (10 `div` 2) / 0
Prolog: X = Y / 0, Y is 10 / 2.
```

(a) Cele scrise în Prolog și Haskell.
(b) Doar cea scrisă în Haskell.
(c) Cele scrise în Racket și Prolog.
(d) Cele scrise în Haskell și Racket.