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. Fie următorul cod Racket:

```
(define (sorted L R)
  (if (null? (cdr L))
      R
      (and (< (car L) (cadr L)) (sorted (cdr L) R))))
(sorted '(55 42 13 6 5) #t)
```

   (a) funcția este recursivă pe coadă iar apelul returnează `#f`
   (b) funcția este recursivă pe stivă iar apelul returnează `#f`
   (c) funcția este recursivă pe stivă iar apelul returnează `#t`
   (d) funcția este recursivă pe coadă iar apelul returnează `#t`

2. Ce complexitate temporală, respectiv spațială (spațiu suplimentar pe stivă), are următoarea implementare Racket, unde n este lungimea listei `L`?

```
(define (f L)
  (cond [(= 1 (length L)) (car L)]
        [(odd? (car L)) (car L)]
        [else (f (cdr L))]))
```

   (a) $O(n^2)$ și $O(1)$
   (b) $O(n)$ și $O(n)$
   (c) $O(n)$ și $O(1)$
   (d) $O(n^2)$ și $O(n)$

3. Ce afișează (prin funcția `display`) codul de mai jos? (fără rezultatul final)

```
(define (f n)
  (cond [(< n 2) 1]
        [else (display n) (+ (f (- n 2)) (f (- n 4)))]))
(f 8)
```

   (a) `8 6 4 2 2 4 2`
   (b) `8 6 4 4 2 2 2`
   (c) `8 6 6 4 4 4 2 2 2 2`
   (d) `8 6 4 2 4 2 2`

4. Ce rezultat are apelul de mai jos?

```
reverse . (foldl (flip (:)) []) $ [0..5]
```

   (a) `[0,1,2,3,4,5]`
   (b) `[5,4,3,2,1,0]`
   (c) `[[5],[4],[3],[2],[1],[0]]`
   (d) `[[0],[1],[2],[3],[4],[5]]`

5. Ce rezultat are apelul de mai jos?

```
(apply map append '(((1 2)) ((3 4))))
```

   (a) `((1 2 3 4))`
   (b) `(1 2 3 4)`
   (c) `((1 3) (2 4))`
   (d) `(1 3 2 4)`

6. Care va fi rezultatul evaluării următoarei expresii Haskell?

```
elem [5]
```

   (a) o funcție
   (b) eroare la compilare
   (c) eroare la rulare
   (d) `5`

7. Care va fi rezultatul evaluării următoarei expresii Racket?

```
(let* ([H (car L)]
       [T (cdr L)]
       [L '(1 5 10 15)])
  (if (null? T)
      H
      T))
```

   (a) Eroare
   (b) `(1)`
   (c) `(5 10 15)`
   (d) `1`

8. Ce va afișa următorul program Racket?

```
(define x 5)
(let ((x 100) (y x))
  (let* ((y (+ y 1)) (x (* 2 y)))
    (+ x y)))
```

   (a) `18`
   (b) `303`
   (c) `301`
   (d) `16`

9. Fie definițiile următoarelor funcții în Racket:

```
(define f (lambda (x) (f x)))
(define (g x) (lambda () (g x)))
```

   Care este diferența dintre apelurile `(f 10)` și `(g 10)`?

   (a) Primul va cicla la infinit, iar al doilea va întoarce o funcție.
   (b) Ambele apeluri vor întoarce o funcție.
   (c) Ambele apeluri vor cicla la infinit.
   (d) Primul va întoarce o funcție, iar al doilea va cicla la infinit.

10. Ce rezultat are codul Haskell de mai jos?

```
(+) ((10/) (-5)) $ (2-) 1
```

   (a) `-1`
   (b) `-2.5`
   (c) `2.5`
   (d) `5`

11. Care este tipul următoarei funcții Haskell:

```
f x y z = x . y $ z
```

   (a) `(b -> c) -> (s -> b) -> s -> c`
   (b) `(b -> s) -> (c -> b) -> s -> c`
   (c) `(b -> c) -> (b -> c) -> b -> c`
   (d) eroare de sinteză

12. Care este tipul următoarei funcții Haskell:

```
f x y z = map (\x -> z (y x)) x
```

   (a) `[a] -> (a -> c) -> (c -> b) -> [b]`
   (b) `c -> (a -> d) -> (d -> b) -> [a]`
   (c) `[c] -> (a -> c) -> (c -> b) -> [a]`
   (d) eroare de sinteză

13. Ce fel de polimorfism utilizează funcția `f`?

```
f x y = x + head y
```

   (a) atât parametric, cât și ad-hoc
   (b) doar ad-hoc
   (c) doar parametric
   (d) funcția nu este polimorfică

14. La ce se evaluează expresia `expr` din următorul cod?

```
data Direction = Left | Right deriving (Eq, Show)
expr = filter id $ map (== Right) $ [Left, Right, Right, Left]
```

   (a) `[True, True]`
   (b) `[Right, Right]`
   (c) Eroare de sinteză de tip

(d) Eroare pentru că operatorul `(==)` nu este supraîncărcat

15. Care expresie Haskell se evaluează la `True`?

(a) `map (+ 1) [1,2,3] == fmap (+ 1) [1,2,3]`
(b) `map (+ 1) (Just 1) == fmap (+ 1) (Just 1)`
(c) `map (+ 1) Nothing == fmap (+ 1) Nothing`
(d) `map (+ 1) (Just 1) == fmap (+ 1) Nothing`

16. Fie următorul cod în Prolog:

```
maximum(X, Y, M) :- X >= Y, M = X ; Y >= X, M = Y.
```

Ce va afișa următoarea interogare?

```
?- maximum(3, 2, 3).
```

(a) `true; false.`
(b) `true; true.`
(c) `true.`
(d) `false.`

17. Ce va afișa următoarea interogare?

```
?- forall(member(X=Y, [3*2=3*2, 6*1=2*3, 4*4=16]), X = Y).
```

(a) `false.`
(b) `true.`
(c) `[3*2=3*2].`
(d) `[3*2=3*2, 6*1=2*3, 4*4=16].`

18. Fie următoarele fapte în cadrul unui program Prolog:

```
a(1). b(1).
a(2). b(2).
a(3). b(3).
```

Ce va afișa următoarea interogare?

```
?- findall(X/Y, (a(X), !, b(Y); (a(Y), b(X))), L).
```

(a) `L = [1/1, 1/2, 1/3].`
(b) `L = [1/1, 2/2, 3/3, 1/1, 2/2, 3/3].`
(c) `L = [1/2, 1/3, 2/1, 2/3, 3/1, 3/2].`
(d) `L = [1/1, 2/2, 3/3].`

19. Se dă următorul program în Prolog:

```
p(L,K) :- p(L, K, 0).
p([], [], _).
p([X|L], [Y|K], Z) :- Y is Z + X, p(L, K, Y).
```

Ce va afișa încercarea de a satisface următorul scop?

```
?- p([1, 4, 3], L).
```

(a) `L = [1, 5, 8].`
(b) `L = 6.`
(c) `L = 14.`
(d) `L = [1, 5].`

20. Care dintre următoarele definiții **NU** vor produce eroare?

```
1. (let ([x 2] [y 2])
     (+ x y))
2. f = let x = 2
           y = 2
           in + x y
3. f = let x = 2
           y = 2
           in (+) x y
4. X = 2, Y = 2, Z = X + Y.
5. X = 2, Y = 2, Z =:= X + Y.
6. Z is X + Y, X = 2, Y = 2.
```

(a) `1, 3 și 4`
(b) `3, 4 și 5`
(c) `2, 3, 4 și 6`
(d) `1, 2, 3, 4 și 5`