

Programowanie L

Egzamin zasadniczy

22 czerwca 2011

Liczba punktów	Ocena
0 – 14	2.0
15 – 17	3.0
18 – 20	3.5
21 – 23	4.0
24 – 26	4.5
27 – 30	5.0

W każdym pytaniu proszę wyraźnie zaznaczyć dokładnie jedną odpowiedź.

Pytanie 1. Rozważmy predykat:

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

Dla jakich kombinacji zmiennych X, Y i Z wywołanie `max(X,Y,Z)` nie zakończy się błędem?

- ☐ a. X, Y i Z są nieukonkretnionymi zmiennymi,
- ☐ b. X i Z są nieukonkretnionymi zmiennymi, a Y jest termem arytmetycznym,
- ☐ c. Y jest nieukonkretnioną zmienną, $X = 1$ i $Z = a$,
- ☒ d. X jest nieukonkretnioną zmienną, $Y = 1$ i $Z = a$,
- ☐ e. X jest nieukonkretnioną zmienną, $Y = 1$ i $Z = 1$.

Pytanie 2. Rozważmy program:

```
parent(adam,marta).  
parent(adam,tomek).
```

```
descendant(X,Y) :-  
    parent(Y,X).  
descendant(X,Y) :-  
    descendant(Z,Y),  
    descendant(X,Z).
```

Jaki będzie wynik zapytania ?- `descendant(adam,X)`.

- ☐ a. niepowodzenie,
- ☐ b. pojedynczy sukces,
- ☐ c. Prolog wygeneruje podstawienia $X = \text{marta}$, $X = \text{tomek}$, a po wymuszeniu nawrotu zapętlili się,
- ☒ d. program zapętlili się.

Pytanie 3. Rozważmy predykat:

```
append3(L1, L2, L3, W) :-  
    append(L1, L2, Lp),  
    append(Lp, L3, W).
```

Jaki będzie wynik zapytania ?- `append3([1], Y, [2], W)`.

- ☐ a. niepowodzenie,
- ☐ b. pojedynczy sukces z odpowiedzią $Y = []$, $W = [1, 2]$,
- ☒ c. nieskończenie wiele sukcesów,
- ☐ d. program zapętlili się.

Pytanie 4. Rozważmy predykat:

```
p([], 0).  
p([H|T], N) :-  
    length(H, M),  
    p(T, K),  
    N is M+K.
```

Jaki będzie wynik zapytania ?- `p([[1,2]],N)`.

- ☐ a. $N = 0$,
- ☐ b. $N = 1$,
- ☒ c. $N = 2$,
- ☐ d. $N = 3$.

Pytanie 5. Jaki jest wynik zapytania ?- `2*2 is 4`.

- ☐ a. pojedynczy sukces,
- ☒ b. niepowodzenie,
- ☐ c. `ERROR: is/2: Arguments are not sufficiently instantiated.`

Pytanie 6. Rozważmy predykat:

```
r(a).  
r(b).  
q(d).  
p(Y) :-  
    r(_),  
    q(Y).
```

Ile odpowiedzi znajdzie Prolog na zapytanie $?- p(X)$.

- ☐ a. 0,
- ☒ b. 1,
- ☒ c. 2,
- ☐ d. nieskończenie wiele.

Pytanie 7. Cel $!$ w definicji predykatu może zostać usunięty, gdyż jego wykonanie nie ma żadnego efektu, jeśli:

- ☐ a. jest pierwszym celem w ciele pierwszej klauzuli,
- ☐ b. jest ostatnim celem w ciele pierwszej klauzuli,
- ☒ c. jest pierwszym celem w ciele ostatniej klauzuli,
- ☐ d. jest ostatnim celem w ciele ostatniej klauzuli,
- ☐ e. żadna z powyższych odpowiedzi nie jest poprawna.

Pytanie 8. Rozważmy predykat:

```
rev([], []).  
rev([H|T], R) :-  
    rev(T, TR),  
    append(TR, [H], R).
```

Jaki będzie wynik zapytania $?- rev([1,2,3], R)$.

- ☒ a. $R = [3,2,1]$,
- ☐ b. $R = [3, [2, [1]]]$,
- ☐ c. $R = [3, [2, [1, []]]]$,
- ☐ d. $R = [[[3], 2], 1]$,
- ☐ e. program zapętli się.

Pytanie 9. Rozważmy predykat:

```
sq(Y,X) :- Y is X*X.
```

Jaki będzie wynik zapytania $?- sq(X,9)$.

- ☐ a. $X = 3$,
- ☒ b. $X = 81$,
- ☐ c. `ERROR: is/2: Arguments are not sufficiently instantiated.`

Pytanie 10. Jaki jest wynik zapytania $?- [], [] = [] \mid V$.

- ☒ a. pojedynczy sukces, przy czym $V = []$,
- ☐ b. pojedynczy sukces, przy czym $V = []$,
- ☐ c. pojedynczy sukces, przy czym V jest nieukonkretnioną zmienną,
- ☐ d. niepowodzenie.

Pytanie 11. Rozważmy program:

```
take(N, [H|T], [H|S]) :-
```

```
    N > 0,
```

```
    N1 is N-1,
```

```
    take(N1, T, S).
```

```
take(0, _, []).
```

```
repeat(N, E, X) :-
```

```
    L = [E|L],
```

```
    take(N, L, X).
```

Jaki będzie wynik zapytania ?- repeat(10,a,X).

☐ a. program zapętli się,

☒ b. pojedynczy sukces:

```
X = [a, a, a, a, a, a, a, a, a, a, a]
```

☐ c. nieskończenie wiele sukcesów:

```
X = [a, a, a, a, a, a, a, a, a, a, a];
```

```
X = [a, a, a, a, a, a, a, a, a, a, a];
```

```
X = [a, a, a, a, a, a, a, a, a, a, a];
```

```
itd.,
```

☐ d. nieskończenie wiele sukcesów:

```
X = [a, a, a, a, a, a, a, a, a, a, a];
```

```
X = [a, a, a, a, a, a, a, a, a, a, a];
```

```
X = [a, a, a, a, a, a, a, a, a, a, a];
```

```
itd.,
```

☐ e. niepowodzenie.

Pytanie 12. Rozważmy program:

```
p.
```

```
p :- 1 is _.
```

```
q.
```

```
q :- q.
```

Jaki będzie wynik zapytania ?- p,q.

☐ a. program zapętli się,

☐ b. niepowodzenie,

☐ c. pojedynczy sukces,

☒ d. nieskończenie wiele sukcesów,

☐ e. sukces, a po wymuszeniu nawrotu błąd „ERROR: is/2: Arguments are not sufficiently instantiated”,

☐ f. błąd „ERROR: is/2: Arguments are not sufficiently instantiated”.

Pytanie 13. Rozważmy program:

```
nat(0).  
nat(N) :-  
    nat(M),  
    N is M+1.
```

Jaki będzie wynik zapytania ?- nat(X), nat(Y), (X,Y)=(1,2).

- ☒ a. program zapętli się,
- ☐ b. niepowodzenie,
- ☐ c. pojedynczy sukces,
- ☐ d. nieskończenie wiele sukcesów,
- ☐ e. sukces, a po wymuszeniu nawrotu błąd „ERROR: is/2: Arguments are not sufficiently instantiated”,
- ☐ f. błąd „ERROR: is/2: Arguments are not sufficiently instantiated”.

Pytanie 14. Rozważmy program:

```
p(X,X) :-  
    write('Yes!').  
  
q(X,Y) :-  
    write('Yes!'),  
    X=Y.
```

Wówczas:

- ☐ a. predykaty p i q są równoważne (w tym sensie, że dla dowolnych argumentów X i Y efekty obliczenia celów p(X,Y) i q(X,Y) będą takie same),
- ☐ b. istnieją argumenty X i Y dla których cel p(X,Y) powiedzie się, zaś cel q(X,Y) zawiedzie,
- ☐ c. istnieją argumenty X i Y dla których cel p(X,Y) zawiedzie, zaś cel q(X,Y) powiedzie się,
- ☒ d. żadne z powyższych stwierdzeń nie jest prawdziwe.

Pytanie 15. Rozważmy program:

```
a --> "1".  
a --> a, a.
```

Jaki będzie wynik zapytania ?- a("111","").

- ☐ a. Undefined procedure: a/2. However, there are definitions for: a/0,
- ☐ b. program zapętli się,
- ☐ c. pojedynczy sukces,
- ☒ d. sukces, a po wymuszeniu nawrotu program zapętli się,
- ☐ e. dwa sukcesy,
- ☐ f. nieskończenie wiele sukcesów,
- ☐ g. niepowodzenie.

Pytanie 16. Rozważmy następujący program w Haskellu:

```
data Expr a = Const a | Op (Expr a) (Expr a)
```

```
eval (Const c) f = c
```

```
eval (Op x y) f = eval x f 'f' eval y f
```

Typem funkcji `eval` jest:

- ☐ a. $(a \rightarrow a) \rightarrow a$
- ☐ b. $(a, a) \rightarrow a$
- ☒ c. $\text{Expr } a \rightarrow (a \rightarrow a \rightarrow a) \rightarrow a$
- ☐ d. $\text{Expr } a \Rightarrow a \rightarrow a \rightarrow a$
- ☐ e. $\text{Expr } a \rightarrow a \rightarrow a$
- ☐ f. $\text{Expr } a \rightarrow ((a, a) \rightarrow a) \rightarrow a$

Pytanie 17. Oto definicja funkcji `reverse`:

```
reverse :: [a] → [a]
```

```
reverse [] = []
```

```
reverse (x:xs) = reverse xs ++ [x]
```

gdzie

```
(++) :: [a] → [a] → [a]
```

```
[] ++ ys = ys
```

```
(x:xs) ++ ys = x : (xs ++ ys)
```

Równość `reverse (reverse xs) = xs`

- ☐ a. zachodzi dla wszystkich list `xs`,
- ☒ b. zachodzi dla niektórych list częściowych `xs`,
- ☐ c. nie zachodzi dla skończonych list `xs`,
- ☐ d. zachodzi dla niektórych nieskończonych list `xs`,
- ☐ e. zachodzi dla wszystkich częściowych list `xs`.

Pytanie 18. Wyrażenie `show (show 42)`

- ☐ a. jest niepoprawne składniowo,
- ☐ b. nie posiada typu,
- ☐ c. ma wartość 42,
- ☐ d. ma wartość "42",
- ☒ e. ma wartość "\"42\"".

Pytanie 19. Oto definicja funkcji `map`:

```
map :: (a → b) → [a] → [b]
```

```
map f [] = []
```

```
map f (x:xs) = f x : map f xs
```

Wyrażenie `map ⊥ []`

- ☐ a. jest równe `⊥`,
- ☒ b. jest równe `[]`,
- ☐ c. jest równe `map ⊥ ⊥`,
- ☐ d. ma typ `a`,
- ☐ e. nie posiada typu.

Pytanie 20. Wynikiem której definicji `ones` nie jest struktura cykliczna?

- ☐ a. `ones = 1 : ones`
- ☒ b. `ones = repeat 1`
`repeat :: a → [a]`
`repeat x = x : repeat x`
- ☐ c. `ones = repeat 1`
`repeat :: a → [a]`
`repeat x = xs where xs = x : xs`
- ☐ d. `ones = iterate (λ x → x) 1`
`iterate :: (a → a) → a → [a]`
`iterate f x = xs where xs = x : map f xs`

Pytanie 21. Które z poniższych wyrażień ma inną wartość niż pozostałe?

- ☐ a. `[(x,y) | x ← xs, y ← ys]`
- ☒ b. `do { y ← ys; x ← xs; return (x,y) }`
- ☐ c. `do { x ← xs; y ← ys; return (x,y) }`
- ☐ d. `concatMap (λ x → map (λ y → (x,y)) ys) xs`

Pytanie 22. Wyrażenie `1 >>= 2`

- ☐ a. ma typ `(Num (a → m b), Monad m) => m b`,
- ☐ b. nie posiada typu,
- ☐ c. ma typ `(Num (m a), Monad m) => m a`,
- ☒ d. ma typ `(Num (m a), Num (a → m b), Monad m) => m b`.

Pytanie 23. Wyrażenie

`(λ (_,_) → 42) (head [])`

- ☐ a. jest niepoprawne składniowo,
- ☐ b. ma wartość 42,
- ☐ c. ma wartość równą wartości wyrażenia

`(λ _ → 42) (head [])`

- ☒ d. kończy się błędem wykonania.

Pytanie 24. Która z poniższych równości nie zachodzi? Zmienne p i r oznaczają komendy, zmienne B , C i D oznaczają ciągi komend, a wyrażenie $C[x := e]$ oznacza ciąg C ze wszystkimi wolnymi wystąpieniami zmiennej x zastąpionymi przez wyrażenie e .

- ☐ a. $\text{do } \{ B; x \leftarrow p; \text{return } x \} = \text{do } \{ B; p \}$
- ☒ b. $\text{do } \{ B; x \leftarrow \text{return } e; C; r \} = \text{do } \{ B; C[x := e]; r \}$
- ☐ c. $\text{do } \{ B; x \leftarrow \text{return } e; C; r \} = \text{do } \{ B; C[x := e]; r[x := e] \}$
- ☐ d. $\text{do } \{ B; x \leftarrow \text{do } \{ C; p \}; D; r \} = \text{do } \{ B; C; x \leftarrow p; D; r \}$

Pytanie 25. Które z poniższych wyrażeń nie posiada typu w Haskellu?

- ☐ a. $(2, \text{tail}, [])$
- ☒ b. $[2, \text{tail}, []]$
- ☐ c. $2 : \text{tail } []$
- ☐ d. $\text{tail } \$ 2 : []$

Pytanie 26. Rozważmy funkcję

```
f [] = []  
f (x:xs) = x : aux xs where aux = aux
```

Jaka jest wartość wyrażenia `head $ f [1..]`?

- ☒ a. 1
- ☐ b. `[1..]`
- ☐ c. `[]`
- ☐ d. obliczenie wartości wyrażenia nie zakończy się.

Pytanie 27. Rozważmy funkcję

```
g xs = [ n | n <- xs, even n, even (n `div` 2) ]
```

Ile elementów ma lista `g [1..100]`?

- ☐ a. 0
- ☒ b. 25
- ☐ c. 50
- ☐ d. 100

Pytanie 28. Rozważmy następującą sygnaturę:

$f :: \text{Integer} \rightarrow \text{Integer} \rightarrow \text{Integer}$

Wówczas f jest funkcją, która

- ☐ a. jako argumenty przyjmuje parę liczb całkowitych i zwraca w wyniku liczbę całkowitą,
- ☒ b. przyjmuje jako argument liczbę całkowitą i zwraca w wyniku funkcję, która przyjmuje jako argument liczbę całkowitą i zwraca w wyniku liczbę całkowitą,
- ☐ c. przyjmuje jako argument funkcję, która przyjmuje jako argument liczbę całkowitą i zwraca w wyniku liczbę całkowitą, i zwraca w wyniku liczbę całkowitą.

Pytanie 29. Rozważmy program:

```
class C a where
  (--) :: a → a → [a]
```

Wskaż zdanie fałszywe:

- ☐ a. Funkcja `--` jest metodą klasy `C`.
- ☒ b. Typem funkcji `--` jest $a \rightarrow a \rightarrow [a]$.
- ☐ c. Jeśli typ `Integer` jest instancją klasy `C`, to wyrażenie `2 -- 5` ma typ `[Integer]`.
- ☐ d. Typem funkcji `--` jest $C\ a \Rightarrow a \rightarrow a \rightarrow [a]$.

Pytanie 30. Najogólniejszym typem wyrażenia `1+2` w Haskellu jest

- ☐ a. `Num`
- ☐ b. `Double`
- ☐ c. `Num a`
- ☐ d. `Int`
- ☒ e. `Num a => a`
- ☐ f. `Integer`

Programowanie L

Egzamin zasadniczy

22 czerwca 2011

Liczba punktów	Ocena
0 – 14	2.0
15 – 17	3.0
18 – 20	3.5
21 – 23	4.0
24 – 26	4.5
27 – 30	5.0

W każdym pytaniu proszę wyraźnie zaznaczyć dokładnie jedną odpowiedź.

Pytanie 1. Rozważmy predykat:

```
p([], 0).  
p([H|T], N) :-  
    length(H, M),  
    p(T, K),  
    N is M+K.
```

Jaki będzie wynik zapytania ?- p([[1,2]],N).

- ☐ a. N = 0,
- ☐ b. N = 1,
- ☒ c. N = 2,
- ☐ d. N = 3.

Pytanie 2. Cel ! w definicji predykatu może zostać usunięty, gdyż jego wykonanie nie ma żadnego efektu, jeśli:

- ☐ a. jest pierwszym celem w ciele pierwszej klauzuli,
- ☐ b. jest ostatnim celem w ciele pierwszej klauzuli,
- ☒ c. jest pierwszym celem w ciele ostatniej klauzuli,
- ☐ d. jest ostatnim celem w ciele ostatniej klauzuli,
- ☐ e. żadna z powyższych odpowiedzi nie jest poprawna.

Pytanie 3. Rozważmy predykat:

```
append3(L1, L2, L3, W) :-  
    append(L1, L2, Lp),  
    append(Lp, L3, W).
```

Jaki będzie wynik zapytania ?- append3([1], Y, [2], W).

- ☐ a. niepowodzenie,
- ☐ b. pojedynczy sukces z odpowiedzią $Y = []$, $W = [1, 2]$,
- ☒ c. nieskończenie wiele sukcesów,
- ☐ d. program zapętli się.

Pytanie 4. Rozważmy predykat:

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

Dla jakich kombinacji zmiennych X, Y i Z wywołanie max(X, Y, Z) nie zakończy się błędem?

- ☐ a. X, Y i Z są nieukonkretnionymi zmiennymi,
- ☐ b. X i Z są nieukonkretnionymi zmiennymi, a Y jest termem arytmetycznym,
- ☐ c. Y jest nieukonkretnioną zmienną, $X = 1$ i $Z = a$,
- ☒ d. X jest nieukonkretnioną zmienną, $Y = 1$ i $Z = a$,
- ☐ e. X jest nieukonkretnioną zmienną, $Y = 1$ i $Z = 1$.

Pytanie 5. Rozważmy program:

```
p.  
p :- 1 is _.
```

```
q.  
q :- q.
```

Jaki będzie wynik zapytania ?- p, q.

- ☐ a. program zapętli się,
- ☐ b. niepowodzenie,
- ☐ c. pojedynczy sukces,
- ☒ d. nieskończenie wiele sukcesów,
- ☐ e. sukces, a po wymuszeniu nawrotu błąd „ERROR: is/2: Arguments are not sufficiently instantiated”,
- ☐ f. błąd „ERROR: is/2: Arguments are not sufficiently instantiated”.

Pytanie 6. Rozważmy program:

```
parent(adam,marta).  
parent(adam,tomek).
```

```
descendant(X,Y) :-  
    parent(Y,X).  
descendant(X,Y) :-  
    descendant(Z,Y),  
    descendant(X,Z).
```

Jaki będzie wynik zapytania ?- descendant(adam,X) .

- ☐ a. niepowodzenie,
- ☐ b. pojedynczy sukces,
- ☐ c. Prolog wygeneruje podstawienia $X = \text{marta}$, $X = \text{tomek}$, a po wymuszeniu nawrotu zapętli się,
- ☒ d. program zapętli się.

Pytanie 7. Rozważmy program:

```
take(N, [H|T], [H|S]) :-  
    N > 0,  
    N1 is N-1,  
    take(N1,T,S).  
take(0,_, []).
```

```
repeat(N,E,X) :-  
    L = [E|L],  
    take(N,L,X).
```

Jaki będzie wynik zapytania ?- repeat(10,a,X) .

- ☐ a. program zapętli się,
- ☒ b. pojedynczy sukces:
 $X = [a, a, a, a, a, a, a, a, a, a]$
- ☐ c. nieskończenie wiele sukcesów:
 $X = [a, a, a, a, a, a, a, a, a, a];$
 $X = [a, a, a, a, a, a, a, a, a, a];$
 $X = [a, a, a, a, a, a, a, a, a, a];$
 itd.,
- ☐ d. nieskończenie wiele sukcesów:
 $X = [a, a, a, a, a, a, a, a, a, a];$
 $X = [a, a, a, a, a, a, a, a, a, a];$
 $X = [a, a, a, a, a, a, a, a, a, a];$
 itd.,
- ☐ e. niepowodzenie.

Pytanie 8. Rozważmy predykat:

```
rev([], []).  
rev([H|T], R) :-  
    rev(T, TR),  
    append(TR, [H], R).
```

Jaki będzie wynik zapytania ?- rev([1,2,3], R).

- ☒ a. R = [3,2,1],
- ☐ b. R = [3, [2, [1]]],
- ☐ c. R = [3, [2, [1, []]]],
- ☐ d. R = [[3], 2], 1],
- ☐ e. program zapętlili się.

Pytanie 9. Rozważmy predykat:

```
r(a).  
r(b).  
q(d).  
p(Y) :-  
    r(_),  
    q(Y).
```

Ile odpowiedzi znajdzie Prolog na zapytanie ?- p(X).

- ☐ a. 0,
- ☒ b. 1,
- ☒ c. 2,
- ☐ d. nieskończenie wiele.

Pytanie 10. Rozważmy program:

```
nat(0).  
nat(N) :-  
    nat(M),  
    N is M+1.
```

Jaki będzie wynik zapytania ?- nat(X), nat(Y), (X,Y)=(1,2).

- ☒ a. program zapętlili się,
- ☐ b. niepowodzenie,
- ☐ c. pojedynczy sukces,
- ☐ d. nieskończenie wiele sukcesów,
- ☐ e. sukces, a po wymuszeniu nawrotu błąd „ERROR: is/2: Arguments are not sufficiently instantiated”,
- ☐ f. błąd „ERROR: is/2: Arguments are not sufficiently instantiated”.

Pytanie 11. Jaki jest wynik zapytania ?- $2*2 \text{ is } 4$.

- ☐ a. pojedynczy sukces,
- ☒ b. niepowodzenie,
- ☐ c. ERROR: is/2: Arguments are not sufficiently instantiated.

Pytanie 12. Jaki jest wynik zapytania ?- $[\], [\] = [\] \mid V$.

- ☒ a. pojedynczy sukces, przy czym $V = [\]$,
- ☐ b. pojedynczy sukces, przy czym $V = [\]$,
- ☐ c. pojedynczy sukces, przy czym V jest nieukonkretnioną zmienną,
- ☐ d. niepowodzenie.

Pytanie 13. Rozważmy program:

```
p(X,X) :-  
    write('Yes!').
```

```
q(X,Y) :-  
    write('Yes!'),  
    X=Y.
```

Wówczas:

- ☐ a. predykaty p i q są równoważne (w tym sensie, że dla dowolnych argumentów X i Y efekty obliczenia celów $p(X,Y)$ i $q(X,Y)$ będą takie same),
- ☐ b. istnieją argumenty X i Y dla których cel $p(X,Y)$ powiedzie się, zaś cel $q(X,Y)$ zawiedzie,
- ☐ c. istnieją argumenty X i Y dla których cel $p(X,Y)$ zawiedzie, zaś cel $q(X,Y)$ powiedzie się,
- ☒ d. żadne z powyższych stwierdzeń nie jest prawdziwe.

Pytanie 14. Rozważmy predykat:

```
sq(Y,X) :- Y is X*X.
```

Jaki będzie wynik zapytania ?- $sq(X,9)$.

- ☐ a. $X = 3$,
- ☒ b. $X = 81$,
- ☐ c. ERROR: is/2: Arguments are not sufficiently instantiated.

Pytanie 15. Rozważmy program:

a --> "1".

a --> a, a.

Jaki będzie wynik zapytania ?- a("111", "").

- ☐ a. Undefined procedure: a/2. However, there are definitions for: a/0,
- ☐ b. program zapętli się,
- ☐ c. pojedynczy sukces,
- ☒ d. sukces, a po wymuszeniu nawrotu program zapętli się,
- ☐ e. dwa sukcesy,
- ☐ f. nieskończenie wiele sukcesów,
- ☐ g. niepowodzenie.

Pytanie 16. Wyrażenie

$(\lambda (_, _) \rightarrow 42) (\text{head } [])$

- ☐ a. jest niepoprawne składniowo,
- ☐ b. ma wartość 42,
- ☐ c. ma wartość równą wartości wyrażenia

$(\lambda _ \rightarrow 42) (\text{head } [])$

- ☒ d. kończy się błędem wykonania.

Pytanie 17. Oto definicja funkcji reverse:

$\text{reverse} :: [a] \rightarrow [a]$

$\text{reverse } [] = []$

$\text{reverse } (x:xs) = \text{reverse } xs ++ [x]$

gdzie

$(++) :: [a] \rightarrow [a] \rightarrow [a]$

$[] ++ ys = ys$

$(x:xs) ++ ys = x : (xs ++ ys)$

Równość $\text{reverse } (\text{reverse } xs) = xs$

- ☐ a. zachodzi dla wszystkich list xs,
- ☒ b. zachodzi dla niektórych list częściowych xs,
- ☐ c. nie zachodzi dla skończonych list xs,
- ☐ d. zachodzi dla niektórych nieskończonych list xs,
- ☐ e. zachodzi dla wszystkich częściowych list xs.

Pytanie 18. Oto definicja funkcji map:

```
map :: (a → b) → [a] → [b]
map f [] = []
map f (x:xs) = f x : map f xs
```

Wyrażenie `map ⊥ []`

- ☐ a. jest równe `⊥`,
- ☒ b. jest równe `[]`,
- ☐ c. jest równe `map ⊥ ⊥`,
- ☐ d. ma typ `a`,
- ☐ e. nie posiada typu.

Pytanie 19. Wynikiem której definicji `ones` nie jest struktura cykliczna?

- ☐ a. `ones = 1 : ones`
- ☒ b. `ones = repeat 1`
 `repeat :: a → [a]`
 `repeat x = x : repeat x`
- ☐ c. `ones = repeat 1`
 `repeat :: a → [a]`
 `repeat x = xs where xs = x : xs`
- ☐ d. `ones = iterate (λ x → x) 1`
 `iterate :: (a → a) → a → [a]`
 `iterate f x = xs where xs = x : map f xs`

Pytanie 20. Wyrażenie `show (show 42)`

- ☐ a. jest niepoprawne składniowo,
- ☐ b. nie posiada typu,
- ☐ c. ma wartość `42`,
- ☐ d. ma wartość `"42"`,
- ☒ e. ma wartość `"\"42\""`.

Pytanie 21. Które z poniższych wyrażeń ma inną wartość niż pozostałe?

- ☐ a. `[(x,y) | x ← xs, y ← ys]`
- ☒ b. `do { y ← ys; x ← xs; return (x,y) }`
- ☐ c. `do { x ← xs; y ← ys; return (x,y) }`
- ☐ d. `concatMap (λ x → map (λ y → (x,y)) ys) xs`

Pytanie 22. Która z poniższych równości nie zachodzi? Zmienne p i r oznaczają komendy, zmienne B , C i D oznaczają ciągi komend, a wyrażenie $C[x := e]$ oznacza ciąg C ze wszystkimi wolnymi wystąpieniami zmiennej x zastąpionymi przez wyrażenie e .

- ☐ a. $\text{do } \{ B; x \leftarrow p; \text{return } x \} = \text{do } \{ B; p \}$
- ☒ b. $\text{do } \{ B; x \leftarrow \text{return } e; C; r \} = \text{do } \{ B; C[x := e]; r \}$
- ☐ c. $\text{do } \{ B; x \leftarrow \text{return } e; C; r \} = \text{do } \{ B; C[x := e]; r[x := e] \}$
- ☐ d. $\text{do } \{ B; x \leftarrow \text{do } \{ C; p \}; D; r \} = \text{do } \{ B; C; x \leftarrow p; D; r \}$

Pytanie 23. Które z poniższych wyrażeń nie posiada typu w Haskellu?

- ☐ a. $(2, \text{tail}, [])$
- ☒ b. $[2, \text{tail}, []]$
- ☐ c. $2 : \text{tail } []$
- ☐ d. $\text{tail } \$ 2 : []$

Pytanie 24. Rozważmy funkcję

```
f [] = []  
f (x:xs) = x : aux xs where aux = aux
```

Jaka jest wartość wyrażenia `head $ f [1..]`?

- ☒ a. 1
- ☐ b. `[1..]`
- ☐ c. `[]`
- ☐ d. obliczenie wartości wyrażenia nie zakończy się.

Pytanie 25. Rozważmy funkcję

```
g xs = [ n | n <- xs, even n, even (n `div` 2) ]
```

Ile elementów ma lista `g [1..100]`?

- ☐ a. 0
- ☒ b. 25
- ☐ c. 50
- ☐ d. 100

Pytanie 26. Wyrażenie `1 >=> 2`

- ☐ a. ma typ $(\text{Num } (a \rightarrow m \ b), \text{Monad } m) \Rightarrow m \ b$,
- ☐ b. nie posiada typu,
- ☐ c. ma typ $(\text{Num } (m \ a), \text{Monad } m) \Rightarrow m \ a$,
- ☒ d. ma typ $(\text{Num } (m \ a), \text{Num } (a \rightarrow m \ b), \text{Monad } m) \Rightarrow m \ b$.

Pytanie 27. Rozważmy następującą sygnaturę:

$f :: \text{Integer} \rightarrow \text{Integer} \rightarrow \text{Integer}$

Wówczas f jest funkcją, która

- ☐ a. jako argumenty przyjmuje parę liczb całkowitych i zwraca w wyniku liczbę całkowitą,
- ☒ b. przyjmuje jako argument liczbę całkowitą i zwraca w wyniku funkcję, która przyjmuje jako argument liczbę całkowitą i zwraca w wyniku liczbę całkowitą,
- ☐ c. przyjmuje jako argument funkcję, która przyjmuje jako argument liczbę całkowitą i zwraca w wyniku liczbę całkowitą, i zwraca w wyniku liczbę całkowitą.

Pytanie 28. Rozważmy następujący program w Haskellu:

```
data Expr a = Const a | Op (Expr a) (Expr a)
```

```
eval (Const c) f = c
```

```
eval (Op x y) f = eval x f 'f' eval y f
```

Typem funkcji `eval` jest:

- ☐ a. $(a \rightarrow a) \rightarrow a$
- ☐ b. $(a, a) \rightarrow a$
- ☒ c. $\text{Expr } a \rightarrow (a \rightarrow a \rightarrow a) \rightarrow a$
- ☐ d. $\text{Expr } a \Rightarrow a \rightarrow a \rightarrow a$
- ☐ e. $\text{Expr } a \rightarrow a \rightarrow a$
- ☐ f. $\text{Expr } a \rightarrow ((a, a) \rightarrow a) \rightarrow a$

Pytanie 29. Rozważmy program:

```
class C a where  
  (--) :: a → a → [a]
```

Wskaż zdanie fałszywe:

- ☐ a. Funkcja `--` jest metodą klasy `C`.
- ☒ b. Typem funkcji `--` jest $a \rightarrow a \rightarrow [a]$.
- ☐ c. Jeśli typ `Integer` jest instancją klasy `C`, to wyrażenie `2 -- 5` ma typ `[Integer]`.
- ☐ d. Typem funkcji `--` jest $C \ a \Rightarrow a \rightarrow a \rightarrow [a]$.

Pytanie 30. Najogólniejszym typem wyrażenia `1+2` w Haskellu jest

- ☐ a. `Num`
- ☐ b. `Double`
- ☐ c. `Num a`
- ☐ d. `Int`
- ☒ e. `Num a => a`
- ☐ f. `Integer`

Programowanie L

Egzamin zasadniczy

22 czerwca 2011

Liczba punktów	Ocena
0 – 14	2.0
15 – 17	3.0
18 – 20	3.5
21 – 23	4.0
24 – 26	4.5
27 – 30	5.0

W każdym pytaniu proszę wyraźnie zaznaczyć dokładnie jedną odpowiedź.

Pytanie 1. Rozważmy program:

```
parent(adam,marta).  
parent(adam,tomek).
```

```
descendant(X,Y) :-  
    parent(Y,X).  
descendant(X,Y) :-  
    descendant(Z,Y),  
    descendant(X,Z).
```

Jaki będzie wynik zapytania ?- descendant(adam,X) .

- ☐ a. niepowodzenie,
- ☐ b. pojedynczy sukces,
- ☐ c. Prolog wygeneruje podstawienia $X = \text{marta}$, $X = \text{tomek}$, a po wymuszeniu nawrotu zapętlili się,
- ☒ d. program zapętlili się.

Pytanie 2. Rozważmy predykat:

```
append3(L1, L2, L3, W) :-  
    append(L1, L2, Lp),  
    append(Lp, L3, W).
```

Jaki będzie wynik zapytania ?- append3([1], Y, [2], W) .

- ☐ a. niepowodzenie,
- ☐ b. pojedynczy sukces z odpowiedzią $Y = []$, $W = [1, 2]$,
- ☒ c. nieskończenie wiele sukcesów,
- ☐ d. program zapętli się.

Pytanie 3. Cel $!$ w definicji predykatu może zostać usunięty, gdyż jego wykonanie nie ma żadnego efektu, jeśli:

- ☐ a. jest pierwszym celem w ciele pierwszej klauzuli,
- ☐ b. jest ostatnim celem w ciele pierwszej klauzuli,
- ☒ c. jest pierwszym celem w ciele ostatniej klauzuli,
- ☐ d. jest ostatnim celem w ciele ostatniej klauzuli,
- ☐ e. żadna z powyższych odpowiedzi nie jest poprawna.

Pytanie 4. Rozważmy predykat:

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

Dla jakich kombinacji zmiennych X , Y i Z wywołanie $\text{max}(X, Y, Z)$ nie zakończy się błędem?

- ☐ a. X , Y i Z są nieukonkretnionymi zmiennymi,
- ☐ b. X i Z są nieukonkretnionymi zmiennymi, a Y jest termem arytmetycznym,
- ☐ c. Y jest nieukonkretnioną zmienną, $X = 1$ i $Z = a$,
- ☒ d. X jest nieukonkretnioną zmienną, $Y = 1$ i $Z = a$,
- ☐ e. X jest nieukonkretnioną zmienną, $Y = 1$ i $Z = 1$.

Pytanie 5. Rozważmy program:

```
p.  
p :- 1 is _.
```

```
q.  
q :- q.
```

Jaki będzie wynik zapytania $?- p, q$.

- ☐ a. program zapętli się,
- ☐ b. niepowodzenie,
- ☐ c. pojedynczy sukces,
- ☒ d. nieskończenie wiele sukcesów,
- ☐ e. sukces, a po wymuszeniu nawrotu błąd „ERROR: is/2: Arguments are not sufficiently instantiated”,
- ☐ f. błąd „ERROR: is/2: Arguments are not sufficiently instantiated”.

Pytanie 6. Rozważmy program:

```
take(N, [H|T], [H|S]) :-  
    N > 0,  
    N1 is N-1,  
    take(N1, T, S).  
take(0, _, []).
```

```
repeat(N, E, X) :-  
    L = [E|L],  
    take(N, L, X).
```

Jaki będzie wynik zapytania ?- repeat(10, a, X).

☐ a. program zapętli się.

☒ b. pojedynczy sukces:

$X = [a, a, a, a, a, a, a, a, a, a]$

☐ c. nieskończenie wiele sukcesów:

$X = [a, a, a, a, a, a, a, a, a, a];$

$X = [a, a, a, a, a, a, a, a, a, a];$

$X = [a, a, a, a, a, a, a, a, a, a];$

itd.,

☐ d. nieskończenie wiele sukcesów:

$X = [a, a, a, a, a, a, a, a, a, a];$

$X = [a, a, a, a, a, a, a, a, a, a, a];$

$X = [a, a, a, a, a, a, a, a, a, a, a, a];$

itd.,

☐ e. niepowodzenie.

Pytanie 7. Rozważmy predykat:

```
p([], 0).  
p([H|T], N) :-  
    length(H, M),  
    p(T, K),  
    N is M+K.
```

Jaki będzie wynik zapytania ?- p([[1,2]], N).

☐ a. $N = 0$,

☐ b. $N = 1$,

☒ c. $N = 2$,

☐ d. $N = 3$.

Pytanie 8. Rozważmy program:

```
p(X,X) :-  
    write('Yes!').
```

```
q(X,Y) :-  
    write('Yes!'),  
    X=Y.
```

Wówczas:

- ☐ a. predykaty p i q są równoważne (w tym sensie, że dla dowolnych argumentów X i Y efekty obliczenia celów $p(X,Y)$ i $q(X,Y)$ będą takie same),
- ☐ b. istnieją argumenty X i Y dla których cel $p(X,Y)$ powiedzie się, zaś cel $q(X,Y)$ zawiedzie,
- ☐ c. istnieją argumenty X i Y dla których cel $p(X,Y)$ zawiedzie, zaś cel $q(X,Y)$ powiedzie się,
- ☒ d. żadne z powyższych stwierdzeń nie jest prawdziwe.

Pytanie 9. Jaki jest wynik zapytania ?- $[[], []] = [[] \mid V]$.

- ☒ a. pojedynczy sukces, przy czym $V = [[]]$,
- ☐ b. pojedynczy sukces, przy czym $V = []$,
- ☐ c. pojedynczy sukces, przy czym V jest nieukonkretnioną zmienną,
- ☐ d. niepowodzenie.

Pytanie 10. Rozważmy predykat:

```
sq(Y,X) :- Y is X*X.
```

Jaki będzie wynik zapytania ?- $sq(X,9)$.

- ☐ a. $X = 3$,
- ☒ b. $X = 81$,
- ☐ c. ERROR: is/2: Arguments are not sufficiently instantiated.

Pytanie 11. Jaki jest wynik zapytania ?- $2*2 \text{ is } 4$.

- ☐ a. pojedynczy sukces,
- ☒ b. niepowodzenie,
- ☐ c. ERROR: is/2: Arguments are not sufficiently instantiated.

Pytanie 12. Rozważmy program:

```
nat(0).  
nat(N) :-  
    nat(M),  
    N is M+1.
```

Jaki będzie wynik zapytania ?- nat(X), nat(Y), (X,Y)=(1,2).

- ☒ a. program zapętlili się,
- ☐ b. niepowodzenie,
- ☐ c. pojedynczy sukces,
- ☐ d. nieskończenie wiele sukcesów,
- ☐ e. sukces, a po wymuszeniu nawrotu błąd „ERROR: is/2: Arguments are not sufficiently instantiated”,
- ☐ f. błąd „ERROR: is/2: Arguments are not sufficiently instantiated”.

Pytanie 13. Rozważmy predykat:

```
rev([], []).  
rev([H|T], R) :-  
    rev(T, TR),  
    append(TR, [H], R).
```

Jaki będzie wynik zapytania ?- rev([1,2,3], R).

- ☒ a. R = [3,2,1],
- ☐ b. R = [3,[2,[1]]],
- ☐ c. R = [3,[2,[1,[]]]],
- ☐ d. R = [[3],2],1],
- ☐ e. program zapętlili się.

Pytanie 14. Rozważmy predykat:

```
r(a).  
r(b).  
q(d).  
p(Y) :-  
    r(_),  
    q(Y).
```

Ile odpowiedzi znajdzie Prolog na zapytanie ?- p(X).

- ☐ a. 0,
- ☒ b. 1,
- ☒ c. 2,
- ☐ d. nieskończenie wiele.

Pytanie 15. Rozważmy program:

a --> "1".

a --> a, a.

Jaki będzie wynik zapytania ?- a("111", "").

- ☐ a. Undefined procedure: a/2. However, there are definitions for: a/0,
- ☐ b. program zapętli się,
- ☐ c. pojedynczy sukces,
- ☒ d. sukces, a po wymuszeniu nawrotu program zapętli się,
- ☐ e. dwa sukcesy,
- ☐ f. nieskończenie wiele sukcesów,
- ☐ g. niepowodzenie.

Pytanie 16. Wyrażenie

$(\lambda (_, _) \rightarrow 42) (\text{head } [])$

- ☐ a. jest niepoprawne składniowo,
- ☐ b. ma wartość 42,
- ☐ c. ma wartość równą wartości wyrażenia

$(\lambda _ \rightarrow 42) (\text{head } [])$

- ☒ d. kończy się błędem wykonania.

Pytanie 17. Oto definicja funkcji reverse:

$\text{reverse} :: [a] \rightarrow [a]$

$\text{reverse } [] = []$

$\text{reverse } (x:xs) = \text{reverse } xs ++ [x]$

gdzie

$(++) :: [a] \rightarrow [a] \rightarrow [a]$

$[] ++ ys = ys$

$(x:xs) ++ ys = x : (xs ++ ys)$

Równość $\text{reverse } (\text{reverse } xs) = xs$

- ☐ a. zachodzi dla wszystkich list xs,
- ☒ b. zachodzi dla niektórych list częściowych xs,
- ☐ c. nie zachodzi dla skończonych list xs,
- ☐ d. zachodzi dla niektórych nieskończonych list xs,
- ☐ e. zachodzi dla wszystkich częściowych list xs.

Pytanie 18. Wynikiem której definicji ones nie jest struktura cykliczna?

- ☐ a. `ones = 1 : ones`
- ☒ b. `ones = repeat 1`
`repeat :: a → [a]`
`repeat x = x : repeat x`
- ☐ c. `ones = repeat 1`
`repeat :: a → [a]`
`repeat x = xs where xs = x : xs`
- ☐ d. `ones = iterate (λ x → x) 1`
`iterate :: (a → a) → a → [a]`
`iterate f x = xs where xs = x : map f xs`

Pytanie 19. Wyrażenie `show (show 42)`

- ☐ a. jest niepoprawne składniowo,
- ☐ b. nie posiada typu,
- ☐ c. ma wartość 42,
- ☐ d. ma wartość "42",
- ☒ e. ma wartość "\"42\"".

Pytanie 20. Oto definicja funkcji `map`:

```
map :: (a → b) → [a] → [b]
map f [] = []
map f (x:xs) = f x : map f xs
```

Wyrażenie `map ⊥ []`

- ☐ a. jest równe `⊥`,
- ☒ b. jest równe `[]`,
- ☐ c. jest równe `map ⊥ ⊥`,
- ☐ d. ma typ `a`,
- ☐ e. nie posiada typu.

Pytanie 21. Które z poniższych wyrażeń ma inną wartość niż pozostałe?

- ☐ a. `[(x,y) | x ← xs, y ← ys]`
- ☒ b. `do { y ← ys; x ← xs; return (x,y) }`
- ☐ c. `do { x ← xs; y ← ys; return (x,y) }`
- ☐ d. `concatMap (λ x → map (λ y → (x,y)) ys) xs`

Pytanie 22. Rozważmy program:

```
class C a where
  (--) :: a → a → [a]
```

Wskaż zdanie fałszywe:

- ☐ a. Funkcja `--` jest metodą klasy `C`.
- ☒ b. Typem funkcji `--` jest $a \rightarrow a \rightarrow [a]$.
- ☐ c. Jeśli typ `Integer` jest instancją klasy `C`, to wyrażenie `2 -- 5` ma typ `[Integer]`.
- ☐ d. Typem funkcji `--` jest $C\ a \Rightarrow a \rightarrow a \rightarrow [a]$.

Pytanie 23. Rozważmy następującą sygnaturę:

```
f :: Integer → Integer → Integer
```

Wówczas `f` jest funkcją, która

- ☐ a. jako argumenty przyjmuje parę liczb całkowitych i zwraca w wyniku liczbę całkowitą,
- ☒ b. przyjmuje jako argument liczbę całkowitą i zwraca w wyniku funkcję, która przyjmuje jako argument liczbę całkowitą i zwraca w wyniku liczbę całkowitą,
- ☐ c. przyjmuje jako argument funkcję, która przyjmuje jako argument liczbę całkowitą i zwraca w wyniku liczbę całkowitą, i zwraca w wyniku liczbę całkowitą.

Pytanie 24. Najogólniejszym typem wyrażenia `1+2` w Haskellu jest

- ☐ a. `Num`
- ☐ b. `Double`
- ☐ c. `Num a`
- ☐ d. `Int`
- ☒ e. `Num a => a`
- ☐ f. `Integer`

Pytanie 25. Rozważmy następujący program w Haskellu:

```
data Expr a = Const a | Op (Expr a) (Expr a)
```

```
eval (Const c) f = c
```

```
eval (Op x y) f = eval x f 'f' eval y f
```

Typem funkcji `eval` jest:

- ☐ a. $(a \rightarrow a) \rightarrow a$
- ☐ b. $(a, a) \rightarrow a$
- ☒ c. $Expr\ a \rightarrow (a \rightarrow a \rightarrow a) \rightarrow a$
- ☐ d. $Expr\ a \Rightarrow a \rightarrow a \rightarrow a$
- ☐ e. $Expr\ a \rightarrow a \rightarrow a$
- ☐ f. $Expr\ a \rightarrow ((a, a) \rightarrow a) \rightarrow a$

Pytanie 26. Rozważmy funkcję

```
f [] = []  
f (x:xs) = x : aux xs where aux = aux
```

Jaka jest wartość wyrażenia `head $ f [1..]`?

- ☒ a. 1
- ☐ b. [1..]
- ☐ c. []
- ☐ d. obliczenie wartości wyrażenia nie zakończy się.

Pytanie 27. Rozważmy funkcję

```
g xs = [ n | n <- xs, even n, even (n `div` 2) ]
```

Ile elementów ma lista `g [1..100]`?

- ☐ a. 0
- ☒ b. 25
- ☐ c. 50
- ☐ d. 100

Pytanie 28. Która z poniższych równości nie zachodzi? Zmienne p i r oznaczają komendy, zmienne B , C i D oznaczają ciągi komend, a wyrażenie $C[x := e]$ oznacza ciąg C ze wszystkimi wolnymi wystąpieniami zmiennej x zastąpionymi przez wyrażenie e .

- ☐ a. $\text{do } \{ B; x \leftarrow p; \text{return } x \} = \text{do } \{ B; p \}$
- ☒ b. $\text{do } \{ B; x \leftarrow \text{return } e; C; r \} = \text{do } \{ B; C[x := e]; r \}$
- ☐ c. $\text{do } \{ B; x \leftarrow \text{return } e; C; r \} = \text{do } \{ B; C[x := e]; r[x := e] \}$
- ☐ d. $\text{do } \{ B; x \leftarrow \text{do } \{ C; p \}; D; r \} = \text{do } \{ B; C; x \leftarrow p; D; r \}$

Pytanie 29. Które z poniższych wyrażeń nie posiada typu w Haskellu?

- ☐ a. `(2, tail, [])`
- ☒ b. `[2, tail, []]`
- ☐ c. `2 : tail []`
- ☐ d. `tail $ 2 : []`

Pytanie 30. Wyrażenie `1 >>= 2`

- ☐ a. ma typ $(\text{Num } (a \rightarrow m \ b), \text{Monad } m) \Rightarrow m \ b$,
- ☐ b. nie posiada typu,
- ☐ c. ma typ $(\text{Num } (m \ a), \text{Monad } m) \Rightarrow m \ a$,
- ☒ d. ma typ $(\text{Num } (m \ a), \text{Num } (a \rightarrow m \ b), \text{Monad } m) \Rightarrow m \ b$.

Programowanie L

Egzamin zasadniczy

22 czerwca 2011

Liczba punktów	Ocena
0 – 14	2.0
15 – 17	3.0
18 – 20	3.5
21 – 23	4.0
24 – 26	4.5
27 – 30	5.0

W każdym pytaniu proszę wyraźnie zaznaczyć dokładnie jedną odpowiedź.

Pytanie 1. Rozważmy program:

a --> "1".

a --> a, a.

Jaki będzie wynik zapytania ?- a("111", "").

- ☐ a. Undefined procedure: a/2. However, there are definitions for: a/0,
- ☐ b. program zapętli się,
- ☐ c. pojedynczy sukces,
- ☒ d. sukces, a po wymuszeniu nawrotu program zapętli się,
- ☐ e. dwa sukcesy,
- ☐ f. nieskończenie wiele sukcesów,
- ☐ g. niepowodzenie.

Pytanie 2. Rozważmy predykat:

sq(Y,X) :- Y is X*X.

Jaki będzie wynik zapytania ?- sq(X,9).

- ☐ a. X = 3,
- ☒ b. X = 81,
- ☐ c. ERROR: is/2: Arguments are not sufficiently instantiated.

Pytanie 3. Rozważmy program:

```
p(X,X) :-  
    write('Yes!').
```

```
q(X,Y) :-  
    write('Yes!'),  
    X=Y.
```

Wówczas:

- ☐ a. predykaty p i q są równoważne (w tym sensie, że dla dowolnych argumentów X i Y efekty obliczenia celów $p(X,Y)$ i $q(X,Y)$ będą takie same),
- ☐ b. istnieją argumenty X i Y dla których cel $p(X,Y)$ powiedzie się, zaś cel $q(X,Y)$ zawiedzie,
- ☐ c. istnieją argumenty X i Y dla których cel $p(X,Y)$ zawiedzie, zaś cel $q(X,Y)$ powiedzie się,
- ☒ d. żadne z powyższych stwierdzeń nie jest prawdziwe.

Pytanie 4. Rozważmy predykat:

```
rev([], []).  
rev([H|T], R) :-  
    rev(T, TR),  
    append(TR, [H], R).
```

Jaki będzie wynik zapytania ?- `rev([1,2,3], R)`.

- ☒ a. $R = [3,2,1]$,
- ☐ b. $R = [3, [2, [1]]]$,
- ☐ c. $R = [3, [2, [1, []]]]$,
- ☐ d. $R = [[3], 2], 1]$,
- ☐ e. program zapętlili się.

Pytanie 5. Rozważmy predykat:

```
p([], 0).  
p([H|T], N) :-  
    length(H, M),  
    p(T, K),  
    N is M+K.
```

Jaki będzie wynik zapytania ?- `p([[1,2]],N)`.

- ☐ a. $N = 0$,
- ☐ b. $N = 1$,
- ☒ c. $N = 2$,
- ☐ d. $N = 3$.

Pytanie 6. Rozważmy program:

```
nat(0).  
nat(N) :-  
    nat(M),  
    N is M+1.
```

Jaki będzie wynik zapytania ?- nat(X), nat(Y), (X,Y)=(1,2).

- ☒ a. program zapętli się,
- ☐ b. niepowodzenie,
- ☐ c. pojedynczy sukces,
- ☐ d. nieskończenie wiele sukcesów,
- ☐ e. sukces, a po wymuszeniu nawrotu błąd „ERROR: is/2: Arguments are not sufficiently instantiated”,
- ☐ f. błąd „ERROR: is/2: Arguments are not sufficiently instantiated”.

Pytanie 7. Rozważmy predykat:

```
r(a).  
r(b).  
q(d).  
p(Y) :-  
    r(_),  
    q(Y).
```

Ile odpowiedzi znajdzie Prolog na zapytanie ?- p(X).

- ☐ a. 0,
- ☒ b. 1,
- ☒ c. 2,
- ☐ d. nieskończenie wiele.

Pytanie 8. Jaki jest wynik zapytania ?- 2*2 is 4.

- ☐ a. pojedynczy sukces,
- ☒ b. niepowodzenie,
- ☐ c. ERROR: is/2: Arguments are not sufficiently instantiated.

Pytanie 9. Rozważmy program:

```
take(N, [H|T], [H|S]) :-  
    N > 0,  
    N1 is N-1,  
    take(N1, T, S).  
take(0, _, []).
```

```
repeat(N, E, X) :-  
    L = [E|L],  
    take(N, L, X).
```

Jaki będzie wynik zapytania ?- repeat(10,a,X).

☐ a. program zapętli się,

☒ b. pojedynczy sukces:

X = [a, a, a, a, a, a, a, a, a, a]

☐ c. nieskończenie wiele sukcesów:

X = [a, a, a, a, a, a, a, a, a, a];

X = [a, a, a, a, a, a, a, a, a, a];

X = [a, a, a, a, a, a, a, a, a, a];

itd.,

☐ d. nieskończenie wiele sukcesów:

X = [a, a, a, a, a, a, a, a, a, a];

X = [a, a, a, a, a, a, a, a, a, a];

X = [a, a, a, a, a, a, a, a, a, a];

itd.,

☐ e. niepowodzenie.

Pytanie 10. Rozważmy program:

p.

p :- 1 is _.

q.

q :- q.

Jaki będzie wynik zapytania ?- p,q.

☐ a. program zapętli się,

☐ b. niepowodzenie,

☐ c. pojedynczy sukces,

☒ d. nieskończenie wiele sukcesów,

☐ e. sukces, a po wymuszeniu nawrotu błąd „ERROR: is/2: Arguments are not sufficiently instantiated”,

☐ f. błąd „ERROR: is/2: Arguments are not sufficiently instantiated”.

Pytanie 11. Jaki jest wynik zapytania ?- [[], []] = [[] | V].

☒ a. pojedynczy sukces, przy czym V = [[]],

☐ b. pojedynczy sukces, przy czym V = [],

☐ c. pojedynczy sukces, przy czym V jest nieukonkretnioną zmienną,

☐ d. niepowodzenie.

Pytanie 12. Rozważmy predykat:

append3(L1, L2, L3, W) :-

append(L1, L2, Lp),

append(Lp, L3, W).

Jaki będzie wynik zapytania ?– `append3([1], Y, [2], W)`.

- ☐ a. niepowodzenie,
- ☐ b. pojedynczy sukces z odpowiedzią `Y = [], W = [1,2]`,
- ☒ c. nieskończenie wiele sukcesów,
- ☐ d. program zapętli się.

Pytanie 13. Cel `!` w definicji predykatu może zostać usunięty, gdyż jego wykonanie nie ma żadnego efektu, jeśli:

- ☐ a. jest pierwszym celem w ciele pierwszej klauzuli,
- ☐ b. jest ostatnim celem w ciele pierwszej klauzuli,
- ☒ c. jest pierwszym celem w ciele ostatniej klauzuli,
- ☐ d. jest ostatnim celem w ciele ostatniej klauzuli,
- ☐ e. żadna z powyższych odpowiedzi nie jest poprawna.

Pytanie 14. Rozważmy predykat:

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

Dla jakich kombinacji zmiennych `X`, `Y` i `Z` wywołanie `max(X,Y,Z)` nie zakończy się błędem?

- ☐ a. `X`, `Y` i `Z` są nieukonkretnionymi zmiennymi,
- ☐ b. `X` i `Z` są nieukonkretnionymi zmiennymi, a `Y` jest termem arytmetycznym,
- ☐ c. `Y` jest nieukonkretnioną zmienną, `X = 1` i `Z = a`,
- ☒ d. `X` jest nieukonkretnioną zmienną, `Y = 1` i `Z = a`,
- ☐ e. `X` jest nieukonkretnioną zmienną, `Y = 1` i `Z = 1`.

Pytanie 15. Wyrażenie

`(λ (_,_) → 42) (head [])`

- ☐ a. jest niepoprawne składniowo,
- ☐ b. ma wartość 42,
- ☐ c. ma wartość równą wartości wyrażenia

`(λ _ → 42) (head [])`

- ☒ d. kończy się błędem wykonania.

Pytanie 16. Rozważmy program:

```
parent(adam,marta).  
parent(adam,tomek).
```

```
descendant(X,Y) :-  
    parent(Y,X).
```

```
descendant(X,Y) :-  
    descendant(Z,Y),  
    descendant(X,Z).
```

Jaki będzie wynik zapytania ?- descendant(adam,X) .

- ☐ a. niepowodzenie,
- ☐ b. pojedynczy sukces,
- ☐ c. Prolog wygeneruje podstawienia $X = \text{marta}$, $X = \text{tomek}$, a po wymuszeniu nawrotu zapętli się,
- ☒ d. program zapętli się.

Pytanie 17. Rozważmy następujący program w Haskellu:

```
data Expr a = Const a | Op (Expr a) (Expr a)
```

```
eval (Const c) f = c
```

```
eval (Op x y) f = eval x f 'f' eval y f
```

Typem funkcji eval jest:

- ☐ a. $(a \rightarrow a) \rightarrow a$
- ☐ b. $(a,a) \rightarrow a$
- ☒ c. $\text{Expr } a \rightarrow (a \rightarrow a \rightarrow a) \rightarrow a$
- ☐ d. $\text{Expr } a \Rightarrow a \rightarrow a \rightarrow a$
- ☐ e. $\text{Expr } a \rightarrow a \rightarrow a$
- ☐ f. $\text{Expr } a \rightarrow ((a,a) \rightarrow a) \rightarrow a$

Pytanie 18. Która z poniższych równości nie zachodzi? Zmienne p i r oznaczają komendy, zmienne B , C i D oznaczają ciągi komend, a wyrażenie $C[x := e]$ oznacza ciąg C ze wszystkimi wolnymi wystąpieniami zmiennej x zastąpionymi przez wyrażenie e .

- ☐ a. $\text{do } \{ B; x \leftarrow p; \text{return } x \} = \text{do } \{ B; p \}$
- ☒ b. $\text{do } \{ B; x \leftarrow \text{return } e; C; r \} = \text{do } \{ B; C[x := e]; r \}$
- ☐ c. $\text{do } \{ B; x \leftarrow \text{return } e; C; r \} = \text{do } \{ B; C[x := e]; r[x := e] \}$
- ☐ d. $\text{do } \{ B; x \leftarrow \text{do } \{ C; p \}; D; r \} = \text{do } \{ B; C; x \leftarrow p; D; r \}$

Pytanie 19. Oto definicja funkcji reverse:

```
reverse :: [a] → [a]
```

```
reverse [] = []
```

```
reverse (x:xs) = reverse xs ++ [x]
```

gdzie

```
(++) :: [a] → [a] → [a]
[] ++ ys = ys
(x:xs) ++ ys = x : (xs ++ ys)
```

Równość `reverse (reverse xs) = xs`

- ☐ a. zachodzi dla wszystkich list `xs`,
- ☒ b. zachodzi dla niektórych list częściowych `xs`,
- ☐ c. nie zachodzi dla skończonych list `xs`,
- ☐ d. zachodzi dla niektórych nieskończonych list `xs`,
- ☐ e. zachodzi dla wszystkich częściowych list `xs`.

Pytanie 20. Które z poniższych wyrażeń nie posiada typu w Haskellu?

- ☐ a. `(2, tail, [])`
- ☒ b. `[2, tail, []]`
- ☐ c. `2 : tail []`
- ☐ d. `tail $ 2 : []`

Pytanie 21. Wyrażenie `1 >=> 2`

- ☐ a. ma typ `(Num (a → m b), Monad m) => m b`,
- ☐ b. nie posiada typu,
- ☐ c. ma typ `(Num (m a), Monad m) => m a`,
- ☒ d. ma typ `(Num (m a), Num (a → m b), Monad m) => m b`.

Pytanie 22. Wynikiem której definicji `ones` nie jest struktura cykliczna?

- ☐ a. `ones = 1 : ones`
- ☒ b. `ones = repeat 1`
 `repeat :: a → [a]`
 `repeat x = x : repeat x`
- ☐ c. `ones = repeat 1`
 `repeat :: a → [a]`
 `repeat x = xs where xs = x : xs`
- ☐ d. `ones = iterate (λ x → x) 1`
 `iterate :: (a → a) → a → [a]`
 `iterate f x = xs where xs = x : map f xs`

Pytanie 23. Oto definicja funkcji `map`:

```
map :: (a → b) → [a] → [b]
map f [] = []
map f (x:xs) = f x : map f xs
```

Wyrażenie `map ⊥ []`

- ☐ a. jest równe `⊥`,
- ☒ b. jest równe `[]`,
- ☐ c. jest równe `map ⊥ ⊥`,
- ☐ d. ma typ `a`,
- ☐ e. nie posiada typu.

Pytanie 24. Wyrażenie `show (show 42)`

- ☐ a. jest niepoprawne składniowo,
- ☐ b. nie posiada typu,
- ☐ c. ma wartość `42`,
- ☐ d. ma wartość `"42"`,
- ☒ e. ma wartość `"\"42\""`.

Pytanie 25. Najogólniejszym typem wyrażenia `1+2` w Haskellu jest

- ☐ a. `Num`
- ☐ b. `Double`
- ☐ c. `Num a`
- ☐ d. `Int`
- ☒ e. `Num a => a`
- ☐ f. `Integer`

Pytanie 26. Rozważmy funkcję

```
f [] = []  
f (x:xs) = x : aux xs where aux = aux
```

Jaka jest wartość wyrażenia `head $ f [1..]`?

- ☒ a. `1`
- ☐ b. `[1..]`
- ☐ c. `[]`
- ☐ d. obliczenie wartości wyrażenia nie zakończy się.

Pytanie 27. Które z poniższych wyrażień ma inną wartość niż pozostałe?

- ☐ a. `[(x,y) | x <- xs, y <- ys]`
- ☒ b. `do { y <- ys; x <- xs; return (x,y) }`
- ☐ c. `do { x <- xs; y <- ys; return (x,y) }`
- ☐ d. `concatMap (λ x → map (λ y → (x,y)) ys) xs`

Pytanie 28. Rozważmy program:

```
class C a where
  (--) :: a → a → [a]
```

Wskaż zdanie fałszywe:

- ☐ a. Funkcja `--` jest metodą klasy `C`.
- ☒ b. Typem funkcji `--` jest `a → a → [a]`.
- ☐ c. Jeśli typ `Integer` jest instancją klasy `C`, to wyrażenie `2 -- 5` ma typ `[Integer]`.
- ☐ d. Typem funkcji `--` jest `C a => a → a → [a]`.

Pytanie 29. Rozważmy następującą sygnaturę:

```
f :: Integer → Integer → Integer
```

Wówczas `f` jest funkcją, która

- ☐ a. jako argumenty przyjmuje parę liczb całkowitych i zwraca w wyniku liczbę całkowitą,
- ☒ b. przyjmuje jako argument liczbę całkowitą i zwraca w wyniku funkcję, która przyjmuje jako argument liczbę całkowitą i zwraca w wyniku liczbę całkowitą,
- ☐ c. przyjmuje jako argument funkcję, która przyjmuje jako argument liczbę całkowitą i zwraca w wyniku liczbę całkowitą, i zwraca w wyniku liczbę całkowitą.

Pytanie 30. Rozważmy funkcję

```
g xs = [ n | n ← xs, even n, even (n `div` 2) ]
```

Ile elementów ma lista `g [1..100]`?

- ☐ a. 0
- ☒ b. 25
- ☐ c. 50
- ☐ d. 100

