

Programowanie 2009 Programming 2009

Egzamin podstawowy Basic exam

30 czerwca 2009 June 30, 2009

Czas trwania egzaminu: 180 minut. Oceny:

Duration of the exam: 180 minutes. Grades:

punkty/points	ocena/grade
0-14	2.0
15-17	3.0
18-20	3.5
21-23	4.0
24-26	4.5
27-30	5.0

Zadanie 1 (1 pkt). Zaprogramuj w Prologu predykat `p(?L)` spełniony wówczas, gdy reszta z dzielenia długości listy `L` przez 4 wynosi 3. Nie wolno używać arytmetyki ani żadnych pomocniczych predykatów.

Problem 1 (1 p). Define in Prolog a predicate `p(?L)` satisfied when the remainder of the division of the length of the list `L` by 4 is 3. You are not allowed to use arithmetic or auxiliary predicates.

```
p([_,_,_]).  
p([_,_,_,_|X):-  
    p(X).
```

Zadanie 2 (1 pkt). Zaprogramuj w Prologu predykat `revapp/3` działający podobnie jak standardowy predykat `append/3` z tą różnicą, że pierwsza lista przed połączeniem z drugą zostaje odwrócona. Mamy więc np.

Problem 2 (1 p). Define in Prolog a predicate `revapp/3` that acts similarly to the standard predicate `append/3` with the difference that the first list is reversed before it is appended to the second one. We have for example:

```
?- revapp([1,2,3],[4,5],X).  
X = [3, 2, 1, 4, 5].
```

Predykat powinien działać poprawnie przynajmniej w trybie `revap(+L1,+L2,?L3)`. Nie wolno używać arytmetyki ani żadnych pomocniczych predykatów.

The predicate should work correctly at least in the mode `revap(+L1,+L2,?L3)`. You are not allowed to use arithmetic or auxiliary predicates.

```
revapp([],X,X).  
revapp([_:_],X,Y):-  
    revapp(_,[_:_],Y).
```

Zadanie 3 (1 pkt). Binarne rozwinięcia liczb nieujemnych reprezentujemy w Prologu w postaci list cyfr 0 i 1 w kolejności od najmniej do najbardziej znaczącej. Zaprogramuj predykat `succ/2` wyznaczający następnik liczby w tej reprezentacji. Mamy więc np.:

```
?- succ([0,1,0,1],L).
```

```
L = [1, 1, 0, 1].
```

```
?- succ([1,1,0,1],L).
```

```
L = [0, 0, 1, 1].
```

```
?- succ([1,1,1,1],L).
```

```
L = [0, 0, 0, 0, 1].
```

Problem 3 (1 p). We represent in Prolog binary representations of nonnegative integers using lists of digits 0 and 1 from the least to the most significant. Define a predicate `succ/2` that computes the successor of a number in this representation. We have for example:

```
succ([], [1]).
succ([0|T], [1|T]).
succ([1|T], [0|S]):-
    succ(T, S).
```

Zadanie 4 (1 pkt). Drzewa binarne o etykietowanych wierzchołkach wewnętrznych reprezentujemy w Prologu w postaci struktur zbudowanych z atomu `leaf/0` i funktora `node/3`, którego pierwszy i trzeci argument są, odpowiednio, lewym i prawym poddrzewem, a drugi — etykietą wierzchołka. Zaprogramuj predykat `r(+T,?N)` spełniony wówczas, gdy etykieta `a/0` występuje `N` razy na każdej ścieżce od korzenia do liścia w drzewie `T`.

Problem 4 (1 p). We represent in Prolog binary trees with labelled internal nodes using structures built from the atom `leaf/0` and the functor `node/3`, whose the first and third argument is, respectively, the left and right subtree, and the second — the label of a node. Define a predicate `r(+T,?N)` satisfied when the label `a/0` occurs `N` times in any path from the root to the leaf in a tree `T`.

```
r(leaf, 0).
r(node(L,a,R), N):-
    !, r(L,N1), r(R,N1), N is N1+1.
r(node(L,_,R), N):-
    r(L,N), r(R,N).
```

Zadanie 5 (1 pkt). Jaka jest odpowiedź maszyny prologowej na zapytanie

Problem 5 (1 p). What is the answer of the Prolog machine to the query

```
?- \+ append(X,Y,Z).
```

```
false.
```

Zadanie 6 (2 pkt). Napisz zbiór produkcji gramatyki bezkontekstowej nad alfabetem terminalnym $\{a, b\}$ i alfabetem nieterminalnym $\{S\}$ generującej język

$$\{w \in \{a, b\}^* : |w|_a \leq |w|_b\},$$

gdzie $|w|_x$ dla $x \in \{a, b\}$ oznacza liczbę wystąpień symbolu x w słowie w .

Problem 6 (2 p). Write the set of productions of a context-free grammar over the terminal alphabet $\{a, b\}$ and nonterminal alphabet $\{S\}$ that generates the language

where $|w|_x$ for $x \in \{a, b\}$ stands for the number of occurrences of a symbol x in a word w .

$$S \rightarrow aSbS \mid bSaS \mid bS \mid \varepsilon$$

Zadanie 7 (1 pkt). Napisz jednym zdaniem definicję pojęcia jednoznacznej gramatyki bezkontekstowej.

Problem 7 (1 p). Write a single sentence containing the definition of the notion of an unambiguous context-free grammar.

Gramatyka jest jednoznaczna, jeżeli każde słowo nad alfabetem tej gramatyki posiada co najwyżej jedno tłumaczenie.

Zadanie 8 (1 pkt). Napisz reguły semantyki operacyjnej wielkich kroków (semantyki naturalnej) dla instrukcji

Problem 8 (1 p). Write the rules of the big-step operational semantics (natural semantics) for the instruction

`repeat c until b`

Wykonanie tej instrukcji polega na cyklicznym wykonywaniu instrukcji c . Pętla jest zakończona, gdy po wykonaniu instrukcji c jest spełniony warunek b .

Execution of this statement consists in repetitive execution of the statement c . The loop is completed when the condition b is satisfied after the execution of the statement c .

$$\frac{\langle c, \pi \rangle \rightarrow \pi' \quad \langle b, \pi' \rangle \rightarrow F}{\langle \text{repeat } c \text{ until } b, \pi \rangle \rightarrow \pi'}$$

$$\frac{\langle c, \pi \rangle \rightarrow \pi' \quad \langle b, \pi' \rangle \rightarrow F \quad \langle \text{repeat } c \text{ until } b, \pi' \rangle \rightarrow \pi''}{\langle \text{repeat } c \text{ until } b, \pi \rangle \rightarrow \pi''}$$

Zadanie 9 (1 pkt). Wyrażenia arytmetyczne języka „While” są opisane następującą gramatyką abstrakcyjną:

n — integer literals
 x — identifiers
 $a ::= n \mid x \mid a_1 \oplus a_2$
 $\oplus ::= + \mid - \mid \times \mid \text{div} \mid \text{mod}$

Podaj reguły semantyki małych kroków (strukturalnej semantyki operacyjnej) dla wyrażeń arytmetycznych języka „While”.

Problem 9 (1 p). Arithmetic expressions of the “While” language are described by the following abstract syntax:

Give the rules of the small-step semantics (structural operational semantics) for arithmetic expressions of the “While” language.

$$\frac{}{\langle n, \pi \rangle \xrightarrow{1} n} \quad \frac{}{\langle x, \pi \rangle \xrightarrow{1} \langle n, \pi \rangle} \quad n = \pi(x)$$

$$\frac{}{\langle a_1, \pi \rangle \xrightarrow{1} \langle a'_1, \pi \rangle}$$

$$\frac{}{\langle a_1 \oplus a_2, \pi \rangle \xrightarrow{1} \langle a'_1 \oplus a_2, \pi \rangle}$$

$$\frac{}{\langle a_2, \pi \rangle \xrightarrow{1} \langle a'_2, \pi \rangle}$$

$$\frac{}{\langle n \oplus a_2, \pi \rangle \xrightarrow{1} \langle n \oplus a'_2, \pi \rangle}$$

$$\frac{}{\langle n_1 \oplus n_2, \pi \rangle \xrightarrow{1} \langle m, \pi \rangle} \quad \text{jeżeli } m = n_1 \oplus n_2$$

Zadanie 10 (3 pkt). Przypomnijmy, że

$\text{foldr} :: (a \rightarrow b \rightarrow b) \rightarrow b \rightarrow [a] \rightarrow b$
 $\text{foldr } _ \ c \ [] = c$
 $\text{foldr } (*) \ c \ (x:xs) = x * \text{foldr } (*) \ c \ xs$

Uzupełnij poniższe definicje funkcji w Haskellu.

Problem 10 (3 p). Recall that

Complete the definitions of the Haskell functions presented below.

`length = foldr (const (1+)) 0`

`(++) = flip $ foldr (:)`

`concat = foldr (++) []`

Zadanie 11 (2 pkt). Podaj definicję klasy Monad.

Problem 11 (2 p). Give the definition of the Monad class.

```
class Monad m where
  (>>=) :: m a -> (a -> m b) -> m b
  (>>)  :: m a -> m b -> m b
  return :: a -> m a
  fail   :: String -> m a
  m >> k == m >>= \_ -> k
  fail = error
```

Niech

Let

```
data Failable a = OK a | Error String
```

Uczyń typ Failable instancją klasy Monad.

Make the type Failable the instance of the Monad class.

```
instance Monad Failable where
  OK a >>= f = f a
  Error s >>= _ = Error s
  return = OK
  fail = Error
```

Zadanie 12 (4 pkt). Wiedząc, że

Problem 12 (4 p). Knowing that

```
(f . g) x = f (g x)
f $ x = f x
flip f a b = f b a
```

wyznacz typy następujących wyrażeń w Haskellu: find the types of the following Haskell expressions:

$(.) (.) (.) :: (a \rightarrow b) \rightarrow (c \rightarrow d \rightarrow a) \rightarrow (c \rightarrow d \rightarrow b)$

$(.) ($) (.) :: (a \rightarrow b) \rightarrow (c \rightarrow a) \rightarrow (c \rightarrow b)$

$($) (.) ($) :: (a \rightarrow b \rightarrow c) \rightarrow (a \rightarrow b \rightarrow c)$

$\text{flip flip flip} :: (a \rightarrow ((b \rightarrow c \rightarrow d) \rightarrow (c \rightarrow d \rightarrow b)) \rightarrow e) \rightarrow (a \rightarrow e)$

Zadanie 13 (2 pkt). Do zbioru wyrażeń arytmetycznych języka „While” dodajemy operatory postinkrementacji i postdekrementacji:

n — integer literals
 x — identifiers
 $a ::= n \mid x \mid a_1 \oplus a_2 \mid a-- \mid a++$
 $\oplus ::= + \mid - \mid \times \mid \text{div} \mid \text{mod}$

Problem 13 (2 p). We add the postincrementation and postdecrementation operators to the set of arithmetic expressions of the “While” language:

błąd, powinno być $x--$
 $x++$

Zdefiniuj dziedzinę denotacji powyższych wyrażeń arytmetycznych.

Define the domain of denotations of the arithmetic expressions above.

$$(\mathbb{Z} \times \Pi) \subseteq \Pi$$

Zdefiniuj semantykę denotacyjną tych wyrażeń arytmetycznych.

Define the denotational semantics of those arithmetic expressions.

$$\llbracket n \rrbracket \pi = \langle n, \pi \rangle$$

$$\llbracket x \rrbracket \pi = \langle \pi(x), \pi \rangle$$

$$\llbracket x++ \rrbracket \pi = \langle \pi(x), \pi[x / \pi(x) + 1] \rangle$$

$$\llbracket x-- \rrbracket \pi = \langle \pi(x), \pi[x / \pi(x) - 1] \rangle$$

$$\llbracket a_1 \oplus a_2 \rrbracket \pi = \langle n_1 \oplus n_2, \pi'' \rangle, \text{ gdzie } \llbracket a_1 \rrbracket \pi = \langle n_1, \pi' \rangle \text{ i } \llbracket a_2 \rrbracket \pi' = \langle n_2, \pi'' \rangle$$

Zadanie 14 (1 pkt). Rozważmy lambda wyrażenia z typami prostymi. Podaj przykład zamkniętego lambda wyrażenia typu $((\alpha \rightarrow \alpha) \rightarrow \beta) \rightarrow \beta$.

Problem 14 (1 p). Consider the simply typed lambda terms. Give an example of a closed lambda term of type $((\alpha \rightarrow \alpha) \rightarrow \beta) \rightarrow \beta$.

$$\lambda f. f(\lambda x. x)$$

Zadanie 15 (1 pkt). Podaj najsłabszy warunek wstępny dla programu

`while N <> 0 do R := 2 * R; N := N - 1; done`

z warunkiem końcowym $R = 2^i$.

Problem 15 (1 p). Give the weakest precondition for the program

with the postcondition $R = 2^i$.

$$N \geq 0 \wedge R \times 2^N = 2^i$$

Zadanie 16 (3 pkt). Udekoruj poniższy program asercjami tak, by otrzymać dowód częściowej poprawności tego programu względem podanej specyfikacji.

$\{N = i\}$

$\{ 1 \times 2^N = 2^i \}$

$R := 1;$

$\{ \}$

$\{ R \times 2^N = 2^i \}$

while $N \neq 0$ do

$\{ R \times 2^N = 2^i \wedge N \neq 0 \}$

$\{ (2 \times R) \times 2^{N-1} = 2^i \}$

$R := 2 * R;$

$\{ R \times 2^{N-1} = 2^i \}$

$N := N - 1;$

$\{ \}$

$\{ R \times 2^N = 2^i \}$

done

$\{ R \times 2^N = 2^i \wedge \neg N \neq 0 \}$

$\{R = 2^i\}$

Problem 16 (3 p). Decorate the program below with assertions to get the proof of partial correctness of this program with respect to the given specification.

Zadanie 17 (3 pkt). Liczby naturalne będziemy reprezentować za pomocą tzw. liczebników Churcha:

Problem 17 (3 p). We will represent natural numbers by so called Church numerals:

gdzie

$$\underline{m} = \lambda f. \lambda x. f^{(m)} x,$$

where

$$\begin{aligned} t^{(0)} s &= s, \\ t^{(n+1)} s &= t(t^{(n)} s), \end{aligned}$$

dla dowolnych termów t i s . Mówimy, że lambda wyrażenie t oblicza funkcję $f : \mathbb{N}^k \rightarrow \mathbb{N}$, jeżeli dla wszelkich liczb naturalnych $m_1, \dots, m_k \in \mathbb{N}$ zachodzi

$$\underline{f(m_1, \dots, m_k)} =_{\beta} t \underline{m_1} \dots \underline{m_k}.$$

Podaj lambda wyrażenia obliczające następnik, dodawanie i mnożenie.

for any terms t and s . We say that a lambda term t computes a function $f : \mathbb{N}^k \rightarrow \mathbb{N}$ if for all natural numbers $m_1, \dots, m_k \in \mathbb{N}$ one has

Give lambda terms computing the successor, addition and multiplication.

$$\text{succ} = \lambda n f x. n f (f x)$$

$$\text{add} = \lambda m n f x. n f (m f x)$$

$$\text{mult} = \lambda n m f. n (m f)$$

Zadanie 18 z powodu błędu w sformułowaniu zostało anulowane.
Wszyscy studenci, którzy przystąpili do egzaminu, otrzymali za nie 1 punkt.