

# Programowanie 2008

## Przykłady pytań egzaminacyjnych

### Część licencjacka

**Zadanie 1 (20 pkt).** Przyjmując, że

```
1, 2 :: (Num t) => t
(*) :: (Num a) => a -> a -> a
sin :: (Floating a) => a -> a
map :: (a -> b) -> [a] -> [b]
```

rozważ poniższe deklaracje w standardowym języku Haskell '98. Podaj opis błędu kompilacji dla tych deklaracji, których kompilacja się nie powiedzie. Dla pozostałych deklaracji podaj typ zadeklarowanej funkcji `f`.

<code>f x = map -1 x</code>	BŁĄD/TYP*:
<code>f x = map (-1) x</code>	BŁĄD/TYP*:
<code>f x = [x] : [1]</code>	BŁĄD/TYP*:
<code>f x = x * sin .1</code>	BŁĄD/TYP*:

\* niepotrzebne skreślić

**Zadanie 2 (20 pkt).** Rozważ poniższe wyrażenia zapisane w standardowym języku Haskell '98. Wskaż wyrażenia, które spowodują błąd kompilacji bądź błąd wykonania i opisz, na czym polegają te błędy. Podaj typy wyrażeń, których kompilacja przebiegnie poprawnie. Podaj wartość wyrażeń, których obliczenie przebiegnie poprawnie.

<code>tail \$ map tail [[],['a']]</code>	BŁĄD KOMPILACJI/WYKONANIA/POPRAWNE*:
<code>let x = x in x x</code>	BŁĄD KOMPILACJI/WYKONANIA/POPRAWNE*:
<code>(\ _ -&gt; 'a') (head [])</code>	BŁĄD KOMPILACJI/WYKONANIA/POPRAWNE*:
<code>(\ (_,_) -&gt; 'a') (head [])</code>	BŁĄD KOMPILACJI/WYKONANIA/POPRAWNE*:

\* niepotrzebne skreślić

**Zadanie 3 (15 pkt).** Przypomnijmy, że

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr _ c [] = c
foldr (*) c (x:xs) = x * foldr (*) c xs
```

```
foldl :: (a -> b -> a) -> a -> [b] -> a
foldl _ c [] = c
foldl (*) c (x : xs) = foldl (*) (c * x) xs
```

```

unfoldr :: (b -> Maybe (a,b)) -> b -> [a]
unfoldr f b = case f b of
  Nothing    -> []
  Just (a,b) -> a : unfoldr f b

```

Podaj typy i wyraż poniższe funkcje w postaci wyrażeń, w których iteracja jest realizowana przez funkcje `foldr`, `foldl` i `unfoldr`.

<pre> f _ [] = [[]] f p (x : xs)     p x = [] : f p xs     otherwise = (x:ys) : zs     where (ys:zs) = f p xs </pre>	<pre> f :: </pre> <pre> f = </pre>
<pre> f n   n == 0 = []       otherwise =       n `mod` 2       : f (n `div` 2) </pre>	<pre> f :: </pre> <pre> f = </pre>
<pre> f [] ys = ys f (x:xs) ys = f xs (x:ys) </pre>	<pre> f :: </pre> <pre> f = </pre>

**Zadanie 4 (25 pkt).** Do prostego języka imperatywnego dodajemy instrukcję pętli

`do  $c_1$  exit when  $b$  otherwise  $c_2$`

Wykonanie tej instrukcji polega na cyklicznym wykonywaniu instrukcji  $c_1$  i  $c_2$ . Iteracja kończy się po instrukcji  $c_1$  w razie spełnienia warunku  $b$ .

a) Instrukcje interpretujemy w zbiorze  $\Pi \hookrightarrow \Pi$  częściowych funkcji przekształcających pamięć w pamięć. Zadać zależność rekurencyjną definiującą funkcję semantyczną dla tej instrukcji.

`[[do  $c_1$  exit when  $b$  otherwise  $c_2$ ]]  $\pi =$`

b) Zadać reguły semantyki operacyjnej wielkich kroków dla tej instrukcji.

c) Zadać reguły semantyki operacyjnej wielkich kroków dla tej instrukcji.

d) Zadać reguły semantyki aksjomatycznej asercji częściowej poprawności dla tej instrukcji.

e) Zadać reguły semantyki aksjomatycznej asercji całkowitej poprawności dla tej instrukcji.

**Zadanie 5 (20 pkt).** Niech typ

```
data Term sig var = Var var | Term sig [Term sig var]
```

reprezentuje terminy nad sygnaturą `sig` ze zbiorem zmiennych `var`. Wprowadź na tym typie strukturę monady, w której operacja `>>=` jest (uogólnionym) podstawieniem.

```
instance Monad (Term sig) where
```

Udowodnij, że zdefiniowane przez Ciebie funkcje spełniają aksjomaty monady.

a) `return x >>= f = f x`

b) `x >>= return = x`

c)  $x \geq (\lambda a \rightarrow f a \geq g) = (x \geq f) \geq g$

## Punktacja

Punkty	Ocena
do 49	2.0
od 50 do 57	3.0
od 58 do 65	3.5
od 66 do 73	4.0
od 74 do 81	4.5
od 82 do 100	5.0

**Uwaga:** Prawdziwy egzamin licencjacki będzie wykazywał  *pewne* podobieństwo do zaprezentowanego powyżej, jednak może zawierać zadania spoza wspomnianych wyżej tematów (w szczególności zadania z programowania w Haskellu nie muszą być w nim aż tak dominujące) i o nieco innej trudności. Proszę traktować ten dokument jako pomoc w przygotowaniu się do egzaminu, a nie zobowiązanie dotyczące zakresu materiału, czy trudności zadań. Z lektury powyższego tekstu proszę też nie wyciągać wniosków odnośnie stopnia przygotowania egzaminu — na przygotowanie powyższego tekstu przeznaczyłem jedynie około 5 godzin, jest więc on pod wieloma względami niedopracowany i nieprzemyślany. Prawdziwy egzamin będzie nieco lepiej przygotowany.

*Tomasz Wierzbicki*