Paradygmaty programowania **IZ, inż. II rok**

Imię i nazwisko	Data	Godzina
Mateusz Grzesiuk	04.11.2020r.	13:15

	1	2	3	4	5	6	7	8	9	10
nr listy: 3 zrobione	V	V	V	V	V					
nr listy: 4 zrobione	V	V	V	V	V					

Lista 3

Zadanie 1

a) Argument „x” w funkcji f1 musi być funkcją, która zwraca wartość dowolnego typu, dla 2 argumentów typu Int. Precyzyjniejsze byłoby jednak stwierdzenie, że „x” jest funkcją, która dla argumentu typu Int, zwraca kolejną funkcję, zwracającą wartość dowolnego typu dla kolejnego argumentu typu Int (Ten sam typ zostanie zwrócony przez f1). Dlatego funkcja f1 jest typu:

   (Int -> Int -> a’) -> a’ = <fun> , gdzie „a’” jest typem wartości zwracanych przez funkcję zwracaną przez „x”.

b) Symbol „^” oznacza konkatenację argumentów typu String, tak więc zarówno „y” jak i „z” muszą być typu String. Implikuje to fakt, iż „x” jest funkcją która zwraca wartość dowolnego typu. Typ argumentu zwracanego przez f2, jest tym samym typem co zwracanego przez „x”. Dlatego funkcja f2 jest typu:

   (string -> 'a) -> string -> string -> 'a = <fun>

Zadanie 2

a) ('a * 'b * 'c -> 'd) -> 'a -> 'b -> 'c -> 'd = <fun>
   Funkcja curry musi pobrać trzy elementy, których typy mogą być dowolne, ale i całkowicie różne od siebie(‘a,’b,’c), stworzyć z nich krotkę, dla której wywoła funkcję, mogącą również zwrócić wartość o zupełnie innym typie (‘d).

b) ('a -> 'b -> 'c -> 'd) -> 'a * 'b * 'c -> 'd = <fun>
Funkcja curry musi pobrać krotkę trzech elementów, których typy mogą być dowolne, ale i całkowicie różne od siebie(‘a,’b,’c), dla wszystkich trzech wywołać funkcję, mogącą również zwrócić wartość o zupełnie innym typie (‘d).

Zadanie 4

Poprawna Funkcja:

```
let rec quicksort = function
 [] -> []
 | [x] -> [x]
 | xs -> let small = List.filter (fun y -> y < List.hd xs ) xs
 and large = List.filter (fun y -> y >= List.hd xs ) (List.tl xs)
      in (quicksort small) @ ((List.hd xs) :: quicksort large);;
```

a) Funkcja quicksort tworzy nieskończoną pętlę, ponieważ stałej „large”, przyporządkowuje listę zawierającą element z którym porównywała wszystkie inne elementy i przez to będzie w kolejnych wywołaniach funkcji porównywać inne elementy tablicy do tego samego elementu. Nie da to żadnych efektów.
b) Funkcja quiksort’ będzie gubić elementy które powtarzają się w tablicy. Zwróci nam posortowaną tablicę bez powtórzeń, co jest niepożądane.

Lista 4

Zadanie 1

a) „x” musi być funkcją przyjmującą 1 argument i zwracającą inną funkcję, przyjmującą kolejny argument. Ta z kolei zwrócona funkcja zwracać może wartość dowolnego typu. Musi tak być, gdyż zaraz po niej użyte są dwa kolejne argumenty. Jest to funkcja o postaci rozwiniętej. „y” i „z” mogą być dowolnych typów.
val f1 : ('a -> 'b -> 'c) -> 'a -> 'b -> 'c = <fun>

b) Funkcja f2 zwraca funkcję od argumentu „z”, który może być dowolnego typu, gdyż nawet raz nie został w niej wykorzystany. Sama funkcja będzie zwracała zawsze to samo, czyli listę elementów dowolnego, ale zarazem jednakowego typu. „y” będzie listą elementów tego typu, a „x” pojedynczym elementem tego typu.
val f2 : 'a -> 'a list -> 'b -> 'a list = <fun>