Lista 1

Zadanie 1 (1p.)

Czy następujące wyrażenia są dobrze typowane? Sprawdź ich typy i wartości w ewaluatorze.

- **if true then 4 else 5**
- **if false then 1 else 3.5**
- **4.75 + 2.34**
- **false || "ab">"cd"**
- **if true then ()**
- **if false then () else 4**
- **let x = 2 in x^"aa"**
- **let y = "abc" in y^y**
- **(fun x -> x.[1]) "abcdef"**
- **(fun x -> x) true**
- **let x = [1;2] in x@x**
- **let rec f f = f+f in f 42**

Zadanie 2 (1p.)

Zidentyfikuj zmienne wolne i związane w wyrażeniach:

- **let x = x in x^x**
- **let x = 10. in let y = x**2. in y*.x**
- **let x = 1 and y = x in x + y**
- **let x = 1 in fun y z -> x*y*z**

Zadanie 3 (1p.)

Jaka jest wartość wyrażenia **f 1** w środowisku otrzymanym po przetworzeniu następujących definicji:

```
let m = 10;;
let f x = m + x;;
let m = 100;;
```

i dlaczego?

Zadanie 4 (1p.)

Napisz wyrażenie, które przekonuje, ze Ocaml używa gorliwej ewaluacji.

Zadanie 5 (1p.)

Anonimową funkcję dodawania dwóch liczb całkowitych możemy zapisać za pomocą wyrażenia **fun x y -> x + y**. Zdefiniuj tę funkcję nadając jej identyfikator **plus** na 3 różne (składniowo) sposoby. Następnie korzystając z jednej z tych definicji napisz funkcję **plus_3** dodawania 3 do dowolnej liczby całkowitej.

Zadanie 6 (1p.)

Jaki jest typ wyrażenia **fun x -> x**? Napisz wyrażenie anonimowe reprezentujące funkcję identycznościową, którego typem w OCamlu jest typ **int -> int**. Napisz wyrażenia typów **('a -> 'b) -> ('c -> 'a) -> 'c -> 'b** oraz **'a -> 'b**.

Zadanie 7 (1p.)

Napisz funkcję definiującą złożenie dwóch funkcji oraz funkcję iterowania wywołania funkcji wykorzystującą funkcję złożenia. Za pomocą iteracji zdefiniuj mnożenie (skorzystaj z faktu, ze operator infiksowy + może być potraktowany jak funkcja dwóch argumentów) i potęgowanie (przy okazji zdefiniuj odpowiedni operator infiksowy).

Zadanie 8 (7p.)

Strumień (tj. nieskończony ciąg) elementów typu **t** możemy reprezentować za pomocą funkcji typu **int -> t** w taki sposób, że dla dowolnej takiej funkcji **s**, **s 0** oznacza pierwszy element strumienia, **s 1** następny, itd. Używając powyższej reprezentacji zdefiniuj następujące funkcje na strumieniach (funkcje te powinny być polimorficzne, tj. powinny działać na strumieniach o dowolnych elementach):

- **hd, tl** - funkcje zwracające odpowiednio głowę i ogon strumienia
- **add** - funkcja dodawania zadanej stałej do każdego elementu strumienia i zwracająca powstały w ten sposób strumień
- **map** - funkcja, która dla zadanej operacji 1-argumentowej przetwarza elementy zadanego strumienia za pomocą tej operacji i zwraca powstały w ten sposób nowy strumień (tak, jak **map** na listach skończonych)
- **map2** - jak wyżej, ale dla zadanych: funkcji 2-argumentowej i 2 strumieni
- **replace** - funkcja, która dla zadanego indeksu **n**, wartości **a** i strumienia **s** zastępuje co **n**-ty element strumienia **s** przez wartość **a** i zwraca powstały w ten sposób strumień
- **take** - funkcja, która dla zadanego indeksu **n** i strumienia **s** tworzy nowy strumień złożony z co **n**-tego elementu strumienia **s**

- **fold** - funkcja, która dla zadanej funkcji **f:'a->'b->'a**, wartości początkowej **a:'a** i strumienia **s** elementów typu **'b** tworzy nowy strumień, którego każdy element jest wynikiem "zwinięcia" początkowego segmentu strumienia **s** aż do bieżącego elementu włącznie za pomocą funkcji **f**, tj. w strumieniu wynikowym element o indeksie **n** ma wartość **(... (f (f a s_0) s_1)... s_n)**
- **tabulate** - funkcja tablicowania strumienia, której wartością powinna być lista elementów strumienia leżąca w zadanym zakresie indeksów.

Zdefiniuj przykładowe strumienie i przetestuj implementację.

W definicji funkcji **tabulate** wykorzystaj możliwość definiowania parametrów opcjonalnych dla funkcji (niech początek zakresu indeksów będzie opcjonalny i domyślnie równy **0**). **Przykład.** Pisząc **let f ?(x=0) y = x + y** deklarujemy, że pierwszy argument funkcji **f** o etykiecie **x** jest opcjonalny, a jego wartość domyślna wynosi **0**. Funkcję **f** można zatem wywołać za pomocą wyrażenia **f 3** (= **3**) lub jawnie podając wartość parametru opcjonalnego, za pomocą składni **f ~x:42 3** (= **45**).