

Programiranje I: 2. izpit

10. februar 2020

Čas reševanja je 150 minut. Veliko uspeha!

1. naloga

a) Napišite funkcijo za izračun skalarne produkta dveh vektorjev iz \mathbb{R}^3 .

```
dot_prod: float * float * float -> float * float * float -> float
```

b) Napišite funkcijo višjega reda, ki sprejme funkcijo dveh argumentov in drugi argument fiksira na podano vrednost.

```
fix_second : ('a -> 'b -> 'c) -> 'b -> 'a -> 'c
```

c) Napišite funkcijo `combine_and_filter` `f xs ys`, kjer imajo argumenti tipe `xs : 'a list`, `ys : 'b list` in `f : ('a -> 'b -> 'c option)` ter vrne vrednost tipa `'c list`. Funkcija isto-
ležne elemente seznamov preslika z `f` in vrne seznam vseh smiselnih rezultatov. Če seznama nista iste dolžine, se izvaja zgolj do konca krajšega od seznamov. Za vse točke naj bo repno rekurzivna.

```
# let safe_minus x y = if x > y then Some (x-y) else None
  combine_and_filter safe_minus [1;0;4;3] [2;1;0;2;5];;
- : int list = [4; 1]
```

d) Napišite funkcijo, ki sprejme predikat in seznam nizov in izpiše z vejico ločene elemente seznama, za katere predikat velja. Pazite, da bo tip funkcije pravilen.

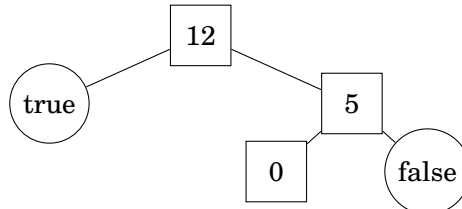
```
# conditional_print;;
- : (string -> bool) -> string list -> unit = <fun>
# let long_enough s = String.length s > 3 in
  conditional_print long_enough ["Ta"; "izpit"; "je"; "neumen!"];;
izpit, neumen!- : unit = ()
```

2. naloga

AB drevesa (ki hranijo dva različna tipa podatkov) definiramo s tipom

```
type ('a, 'b) tree =  
  | Empty  
  | ANode of ('a, 'b) tree * 'a * ('a, 'b) tree  
  | BNode of ('a, 'b) tree * 'b * ('a, 'b) tree
```

a) Definirajte AB drevo `test : (int, bool) tree`, ki predstavlja spodnje drevo.



b) Definirajte funkciji `adepth` in `bdepth` tipa `('a, 'b) tree -> int`, ki vrneti globino najglobljega A oz. B vozlišča. Na zgornjem primeru sta torej obe globini enaki 3, če pa vozlišče, ki vsebuje 0, nadomestimo z novim vozliščem `false`, bi funkcija `adepth` sedaj vrnila 2.

c) Definirajte zapisni tip `result` in funkcijo `count : ('a, 'b) tree -> result`, ki prešteje število posameznih vozlišč v AB drevesu kot je prikazano v primeru.

```
# count test;;  
- : result = {aNodes = 3; bNodes = 2}
```

d) Definirajte funkcijo `is_typemirror : ('a, 'b) tree -> ('b, 'a) tree -> bool`, ki preveri, ali sta si drevesi zrcalni v uporabi A in B vozlišč (torej sta matematično gledano enaki, le da prvo drevo uporablja A vozlišča na mestih, kjer drugo drevo uporablja B vozlišča in obratno).

```
# is_typemirror (ANode (Empty, 1, Empty)) (BNode (Empty, 1, Empty));;  
- : bool = true
```

e) Napišite funkcijo `foldmap fa fb acc tr` za hkratno zlaganje in preslikanje AB dreves. Funkciji `fa : 'c -> 'a -> 'c * 'd` in `fb : 'c -> 'b -> 'c * 'e` sprejmeta akumulator tipa `'c` in vrednost vozlišča, ter vrneti posodobljen akumulator in novo vrednost vozlišča. Funkcija `foldmap` zloži ti dve funkciji preko AB drevesa `tr : ('a, 'b) tree` z začetnim akumulatorjem `acc : 'c`. Kot rezultat vrne par, končno stanje akumulatorja in novo drevo tipa `('d * 'e) tree`, ki vsebuje posodobljene vrednosti vozlišč. Za poenostavitev problema predpostavite, da vrstni red sprehoda po drevesu ni pomemben.

```
# foldmap test (fun acc x -> (acc+x, 0)) (fun acc b -> (acc-1, ())) 0;;  
- : int * (int, unit) tree =  
(15,  
  ANode (BNode (Empty, (), Empty), 0,  
    ANode (ANode (Empty, 0, Empty), 0, BNode (Empty, (), Empty)))
```

3. naloga

Nalogo lahko rešujete v Pythonu ali OCamlu.

Napišite funkcijo `f(k, n)`, ki vrne število vseh zaporedij naravnih števil (naravna števila vsebujejo 0) dolžine `n`, ki se začnejo z 0 in v katerih je razlika med zaporednima členoma manjša ali enaka `k`.