Programiranje I: 1. Izpit

28. januar 2020

Čas reševanja je 150 minut. Veliko uspeha!

1. naloga

a) Napišite funkcijo `option_sum: int option -> int option -> int option`. Funkcija vrne vsoto argumentov, če oba argumenta vsebujeta število, in `None` sicer.

```
# option_sum (Some 1) None;;
- : int option : None
```

b) Napišite funkcijo `twostep_map f l r x`, kjer imajo argumenti tipe `f: ('a -> 'b * 'c)`, `l: ('b -> 'd)`, `r: ('c -> 'e)` in `x : 'a`, rezultat funkcije pa je tipa `'d * 'e`. Funkcija element `x` s funkcijo `f` preslika v par in na komponentah ustrezno uporabi funkciji `l` in `r`.

```
  # twostep_map (fun x -> (x, x)) ((+)1) ((-)2) 3;;
  - : int * int = (4, -1)
```

c) Funkcija `function_repeat: ('a -> int) -> 'a list -> 'a list`, sprejme funkcijo `f` in seznam `list` ter vrne nov seznam, kjer se vsak element `x` seznama `list` ponovi `f x` krat. Nepozitivno število ponovitev pomeni, da elementa ne vključimo v končni seznam. Za vse točke naj bo funkcija repno rekurzivna, kar tudi argumentirajte v komentarju.

```
# function_repeat (fun x -> x) [0;1;2;1;(-2)];;
- : int list = [1;2;2;1]
```

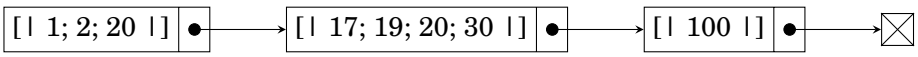
d) Definirajte funkcijo `iterate: ('a -> 'a) -> ('a -> bool) -> 'a -> 'a`, ki sprejme funkcijo `f`, zaustavitveni pogoj in začetno vrednost. Nato funkcijo `f` zaporedoma uporablja, dokler za rezultat ne velja zaustavitveni pogoj. Funkcija naj vrne prvi rezultat, pri katerem zaustavitveni pogoj vrne `true`.

```
  # iterate (fun x -> x *. x) (fun x -> x > 12345.) 2.;;
  - : float = 65536.
```

2. naloga

Napreden povezan seznam je podoben vgrajenemu seznamu v OCaml-u, le da v vozliščih namesto vrednosti hrani tabelo vrednosti (velikosti tabel niso nujno enake). Tako kot običajen povezan seznam je sestavljen iz dveh različnih gradnikov: praznega seznama in vozlišča, ki vsebuje tabelo in preostanek naprednega seznama.

a) Definirajte polimorfen tip `'a improved_list` ter seznam `test : int improved_list`, ki predstavlja spodnji izboljšan seznam:



b) Definirajte funkcijo `ilist_len: 'a improved_list -> int`, ki vrne dolžino podanega seznama.

c) Definirajte funkcijo `get_el: int -> 'a improved_list -> 'a option`, ki vrne i-ti element če ga seznam vsebuje. Za vse točke mora biti funkcija repno rekurzivna.

```
# index test 5;;
- : int option = Some 20
```

d) Definirajte funkcijo `is_sorted: 'a improved_list -> bool`, ki ugotovi ali je napreden seznam urejen (predpostavimo, da vsebuje elemente, ki jih lahko primerjamo z <). Za vse točke mora biti funkcija repno rekurzivna in imeti linearno časovno zahtevnost.

```
# is_sorted test;;
- : bool = false
```

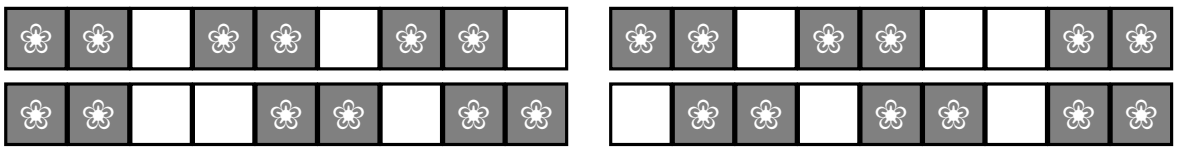
e) Napišite funkcijo `update: 'a improved_list -> int -> 'a -> 'a improved_list`, ki vrne nov napreden seznam, kjer vrednost na indeksu (drugi argument) nadomesti s podano vrednostjo (tretji argument). Pazite, da pri tem začetni seznam ostane nespremenjen. Za vse točke mora biti funkcija repno rekurzivna, kar v komentarju tudi argumentirajte.

```
# index (update test 5 (-3)) 5;;
- : int option = Some (-3)
# index test 5;;
- : int option = Some 20
```

3. naloga

a) Na mizo dolžine n želimo za dekoracijo postaviti m posod za rože, kjer je vsaka posoda dolžine l. Posode za rože postavljamo eno za drugo, med dvema zaporednima posodama pa mora biti vsaj 1 enota mize prazna. Sestavite funkcijo, ki sprejme n, m in l ter vrne število vseh različnih postavitev posod za rože na mizo. Postaviti moramo vse posode, hkrati pa posod med seboj ne razlikujemo.

Primer za vse 4 možne postavitve pri mizi dolžine 9 s tremi posodami dolžine 2:



b) Sedaj imamo korita različnih dolžin. Sestavite funkcijo, ki sprejme dolžino mize n in seznam celih števil, ki predstavlja dolžine posod za rože, in vrne število različnih postavitev posod na mizo. Pri tem je vrstni red korit določen z vrstnim redom dolžin v podanem seznamu (med koriti iste dolžine ne razlikujemo).