

Programiranje I: 2. izpit

25. avgust 2020

Čas reševanja je 150 minut. Veliko uspeha!

1. naloga

a) Napišite funkcijo za izračun kota med dvema ravninskima vektorjema.

```
angle_between: float * float -> float * float -> float
```

b) Napišite funkcijo, ki pretvarja sezname primerne dolžine v trojice.

```
list_to_triple : 'a list -> ('a * 'a * 'a) option
```

c) Definirajte zapisni tip counter s celoštevilskimi polji lt, eq in gt. Funkcija compare_with sprejme seznam in vrednost in vrne koliko elementov seznama je manjših, enakih oziroma večjih kot podana vrednost. Rezultat naj bo tipa counter, za vse točke pa naj se funkcija po seznamu sprehodi zgolj enkrat.

```
compare_with : 'a list -> 'a -> counter
```

d) Napišite funkcijo, ki sestavi kompozitum seznama funkcij.

```
apply_all : ('a -> 'a) list -> 'a -> 'a
```

Kot primer, apply_all [f1; f2; f3] vrne funkcijo, ki x preslika v f1(f2(f3(x))). Za vse točke naj bo kompozitum sestavljen tako, da med izvajanjem ne pride do stack overflow napake. Kot test lahko uporabite:

```
let long_test = List.init 1000000 (fun _ -> (+) 1) in  
apply_all long_test 0
```

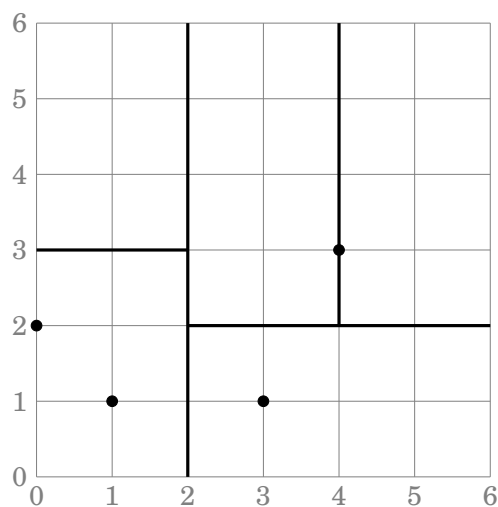
2. naloga

Za razvrščanje točk v ravnini uporabljamo *delilna drevesa*. Vozlišča drevesa razdelijo ravnino (po x ali y koordinati), v listih pa se nahaja seznam vseh elementov v podravnini.

```
type xytree =  
  | Xsplit of int * xytree * xytree  
  | Ysplit of int * xytree * xytree  
  | Elements of (int * int) list
```

Vozlišče `Xsplit (2, lt, rt)` ravnino razdeli glede na premico $x = 2$. V poddrevesu `lt` se nahajajo točke s prvo koordinato *manjšo ali enako* 2, v `rt` pa točke s prvo koordinato večjo od 2. Vrstni red točk v seznamih ni pomemben.

a) Definirajte primer delilnega drevesa `example`, ki ga prikazuje slika. Točke naj bodo vstavljene v pravilno podravnino.



b) Napišite funkcijo `num_of_elements`, ki prešteje število točk, ki se nahajajo v delilnem poddrevesu.

```
# num_of_elements example;;  
- : int = 4
```

c) Definirajte funkcijo `insert` za vstavljanje točk v delilno drevo.

d) Definirajte funkcijo `alternates`, ki preveri ali delilno poddrevo izmenično uporablja delilno koordinato. To pomeni, da če na prvem koraku ravnino delimo glede na x -os, jo v drugem glede na y in tako naprej.

e) Napišite funkcijo `boxed_correctly`, ki preveri ali so vsi elementi v delilnem drevesu v pravilnem vozlišču.

3. naloga

Nalogo lahko rešujete v Pythonu ali OCamlu.

Parnost permutacije π definiramo kot parnost števila njenih *inverzij*, torej takih parov števil $i < j$, da velja $\pi_i > \pi_j$. Na primer, identična permutacija je soda, ker nima inverzij, permutacija

$$\rho = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$$

pa je liha, saj ima tri inverzije: $\rho_1 > \rho_4$, $\rho_2 > \rho_4$ in $\rho_3 > \rho_4$. Enostavno lahko preverimo (pa tudi pri *Algebri 1* ste se učili), da:

- je kompozitum dveh sodih ali dveh lihih permutacij soda permutacija,
- je kompozitum sode in lihe permutacije liha permutacija,
- je cikel $(123 \dots n)$ liha permutacija natanko tedaj, kadar je n sod.

Napišite funkcijo *soda*, ki sprejme naravno število n in ob vsakem klicu vrne naključno izbrano *sodo* permutacijo prvih n naravnih števil. Funkcija naj deluje v času $O(n)$, porabi naj največ $O(n)$ dodatnega prostora, vsaka permutacija pa naj se pojavi z enako verjetnostjo (ki je za $n < 2$ enaka 1, sicer pa $\frac{2}{n!}$). Pravilnost rešitve utemeljite v komentarjih.