Programiranje I: 1. izpit

27. junij 2019

Čas reševanja je 150 minut. Veliko uspeha!

1. naloga

Definirajte tip, ki predstavlja kompleksna števila izražena v kartezičnih koordinatah:

```
type complex = { re : float ; im : float }
```

a) Napišite funkcijo, ki sešteje dve kompleksni števili.

```
complex_add : complex -> complex -> complex
```

b) Napišite funkcijo, ki izračuna konjugacijo kompleksnega števila.

```
complex_conjugate : complex -> complex
```

c) Napišite funkcijo

```
list_apply_either : ('a -> bool) -> ('a -> 'b) -> ('a -> 'b) -> 'a list -> 'b list
```

ki sprejme logični predikat `pred`, funkciji `f` in `g`, in seznam `xs`. Za vsak element `x` seznama `xs` izračuna vrednost `f`, če `x` zadošča predikatu `pred`, sicer pa izračuna vrednost `g`. Na koncu vrne seznam vseh tako dobljenih rezultatov.

d) Želimo izračunati vrednost polinoma v neki točki. Za poenostavitev se bomo omejili zgolj na cela števila. Polinome bomo predstavili kot seznam koeficientov (razvrščene naraščajoče glede na pripadajočo potenco), na primer $x^3 - 2x + 3$ predstavimo z `p = [3; -2; 0; 1]`. Napišite funkcijo

```
eval_poly : int list -> int -> int
```

ki sprejme seznam koeficientov in točko, v kateri naj izračuna vrednost polinoma. Če uporabimo `p` iz zgornjega primera, `eval_poly p 0 = 3`, and `eval_poly p 3 = 24`. **Funkcija naj bo repno rekurzivna.**

2. naloga

Društvo slovenskih vrtičkarjev nas je najelo, da jim pomagamo voditi evidenco. Vrt je razdeljen v več vrtičkov, ki so bodisi dodeljeni določenemu najemniku ali pa prosti. Vsak najemnik se lahko odloči, ali svoj del vrta obdeluje ali pa svoj del vrta razdeli na manjše kose, ki jih nato oddaja drugim vrtičkarjem.

Društvo zahteva, da v primeru oddajanja v najem, vrta ne razdelimo na 0 delov. Temu primerno bomo za modeliranje uporabili tip `vrt * vrt list`, torej par oblike

$$(\text{obvezni del}, \text{preostali deli})$$

Če je vrt oddan zgolj enemu lastniku, je desna komponenta para prazen seznam in vrstni red delov ni pomemben.

Za modeliranje vrtičkov uporabimo tip

```
type najemnik = string

type vrt = Obdelovan of lastnik
         | Oddan of lastnik * (vrt * vrt list)
         | Prost
```

a) Definirajte `vrt_primer`, ki modelira `vrt` z lastnostmi:
Najemnica Kovalevskaya je razdelila svoj vrt na tri dele, kjer

- najemnik Galois svoj del obdeluje
- najemnik Lagrange svoj del obdeluje
- preostali del je prost

b) Napišite funkcijo

```
obdelovalec_vrta : vrt -> najemnik option
```

ki vrne ime najemnika, ki *direktno* obdeluje vrt (torej ga ne oddaja v najem), če obstaja. Na primer, `obdelovalec_vrta vrt_primer = None`.

c) Napišite funkcijo

```
globina_oddajanja : vrt -> int
```

ki izračuna maksimalno verigo deljenja in oddajanja delov vrta. Na zgornjem primeru torej vrne 1, saj vrt oddaja zgolj Kovalevskaya.

d) Napišite funkcijo

```
v_uporabi : vrt -> bool
```

ki preveri, ali obstaja del vrta, ki ga kdo obdeluje. V nasprotnem primeru je vrt morda deljen na več delov, vendar so vsi deli neobdelani.

e) Napišite funkcijo

```
vsi_najemniki : vrt -> najemnik list
```

ki vrne seznam vseh najemnikov, ki so na katerikoli način vključeni v vrt. Vrstni red ni pomemben.

f) Napišite funkcijo

```
vsi_obdelovalci : plot -> najemnik list
```

ki vrne seznam vseh najemnikov, ki obdelujejo kakšen del vrt. Na zgornjem primeru torej vrne `["Galois"; "Lagrange"]`. Vrstni red ponovno ni pomemben.

3. naloga

Nalogo lahko rešujete v Pythonu ali OCamlu.

Tovorne ladje natovarjamo z različno velikimi zabojniki. Vsak zabojnik ima natanko določeno težo, na primer 1, 3, 4, 7 ali 10 ton. Prav tako ima vsaka ladja določeno nosilnost, na primer 5, 40 ali 300 ton. Vaša naloga je, da izračunate število različnih načinov kako natovorimo ladjo do natanko polne nosilnosti.

Vsak tip zabojnika lahko izberemo poljubno mnogokrat in vrstni red natovarjanja ne igra nobene vloge. Zato predpostavimo, da zabojnike vedno natovorimo v točno določenem vrstnem redu, kjer vedno najprej natovorimo lažje zabojnike, in se nato pomaknemo proti težjim.

Kot primer, če ima naša ladja nosilnost 5 ton in imamo zabojnike s težo 1, 3, 4, 7 in 10 ton, potem imamo natanko tri možna natovarjanja.

- 5 zabojnikov s težo 1 tona
- 2 zabojnika teže 1 in en zabojnik teže 3
- 1 zabojnik teže 1 in en zabojnik teže 4

Za lažje reševanje si definirajte nabor tež kot konstanto

```
zabojniki = [1, 3, 4, 7, 10]
```

Napišite funkcijo, ki kot vhod sprejme nosilnost ladje in izračuna število različnih načinov, kako ladjo natovorimo z prej podanimi zabojniki.

Funkcija naj bo dovolj učinkovita, da lahko izračuna rezultat za ladjo z nosilnostjo 300 ton.

Za dodatne točke napišite funkcijo, ki rešitve ne zgolj prešteje, temveč tudi vrne. (Te raje ne testirajte na ogromnih primerih.)