Programiranje I: 2. izpit

7. februar 2022

Čas reševanja je 120 minut. Veliko uspeha!

1. naloga

a) Napišite predikat `je_idempotent : (int * int) * (int * int) -> bool`, ki za dano dvodimenzionalno matriko pove, ali je idempotentna. Matrika P je idempotentna, kadar velja $P^2 = P$.

Primera dveh idempotentnih matrik:

```
# let identiteta = ((1, 0), (0, 1));;
# let test1 = ((3, -6), (1, -2));;
# je_idempotent identiteta;;
- : bool = true
# je_idempotent test1;;
- : bool = true
```

Primer matrike, ki ni idempotentna:

```
let test2 = ((1, -6), (1, -2));;
# je_idempotent test2;;
- : bool = true
```

b) Napišite funkcijo, ki izračuna produkt neničelnih elementov v seznamu. Produkt praznega seznama naj bo enak 1. Funkcija naj bo repno rekurzivna.

c) Napišite funkcijo `stalin`, ki iz danega seznama odstrani vse elemente, ki niso v strogo naraščajočem vrstnem redu, po požrešni metodi.

Primer:

```
# stalin [6; 3; 10; 5; 16];;
- : int list = [6; 10; 16]
```

d) Napišite funkcijo `splosni_collatz` ki za dani funkciji `f` in `g`, `pogoj : int -> bool`, začetno število `z`, končno število `k` in maksimalno število ponovitev `n`, računa člene splošne verzije Collatzovega zaporedja, dokler ne pride do števila `k` ali do maksimalnega števila ponovitev. Člen zaporedja je določen s prejšnjim členom tako, da se uporabi funkcija `f` na prejšnjem členu, če zanj velja pogoj, sicer pa se uporabi funkcija `g`.

Primer: originalno Collatzovo zaporedje

```
# let pogoj x = (x mod 2) = 0;;
# let f x = x / 2;;
# let g x = 3 * x + 1;;
# splosni_collatz f g pogoj 12 1 1000;;
- : int list = [12; 6; 3; 10; 5; 16; 8; 4; 2; 1]
```

V primeru začnemo s številom 12, ki je deljivo z 2 (pogoj), zato uporabimo funkcijo `f`. Dobimo število 6, ki je spet deljivo z 2, zato spet uporabimo `f`. Naslednje število je 3, ki pa ni deljivo z 2, zato uporabimo funkcijo `g`, in dobimo število 10. Postopek ponavljamo, dokler ne pridemo do končnega števila 1.

2. naloga

Računanje aritmetičnih izrazov lahko predstavimo s skladom, kjer aritmetični izraz predstavimo kot zaporedje števil in operatorjev na skladu (seznamu). Izraz `(3 + 4) / 5` predstavimo kot: `"3 4 + 5 /"`. Aritmetične izraze v taki obliki evalviramo od leve proti desni tako, da najprej s sklada vzamemo oba argumenta in operator, operator apliciramo na operanda in na sklad vrnemo rezultat. To počnemo toliko časa, dokler ni na skladu zgolj en element, ki je končni rezultat. V kolikor med izvajanjem na vrhu sklada ni pravilnih oblik elementov, ali na koncu ostane nepravilno število elementov je tekom izvajanja prišlo do napake.

```
type 'a operator = 'a -> 'a -> 'a

type 'a atom = Value of 'a | Operator of 'a operator

type 'a term = 'a atom list

type 'a result = Finished of 'a | Term of 'a term | Error

let plus = Operator ( + )

let krat = Operator ( * )

let deljeno = Operator ( / )

let primer : int term = [ Value 3; Value 4; plus; Value 5; deljeno ]
```

a) Definirajte izraza, ki ju predstavljata sledeča niza: `"1 2 + 4 - 5 *"`, `"5.3 4.6 /. 1.7 *."`. Definirajte tudi poljuben izraz, ki predstavlja neveljaven izraz (kjer bi med izvajanjem prišlo do napake)

b) Definiraje funkcijo `korak 'a term -> 'a result`, ki sprejme izraz (sklad) in poskuša narediti en korak evalvacije. Če podan izraz predstavlja končni rezultat naj vrne obliko `Finished`, če je mogoče narediti en korak naj naredi en korak in vrne stanje po opravljenem koraku v obliki `Term`, v nasprotnem primeru pa naj vrne obliko `Error`.

```
# korak primer;;
- : int result = Term [Value 7; Value 5; Operator <fun>]
# korak [Value 7; Value 5; deljeno];;
- : int result = Term [Value 1]
```

c) Definirajte funkcijo `izvedi : 'a term -> 'a option`, ki sprejme izraz in ga evalvira. Če kjerkoli med izvajanje pride do napake naj funkcija vrne `None`.

d) Definirajte predikat `valid : 'a term -> bool`, ki sprejme izraz in preveri, ali je podan izraz veljaven (ali bi izvedba izraza v celoti pripeljala do rezultata, ali bi vmes prišlo do napake). Za vse točke naj bo funkcija repno rekurzivna in ne gre v celoti izvesti izraza (ne aplicira funkcij).

e) Očitno lahko iz seznama operatorjev in vrednosti pripravimo izraz. Vaša naloga je pripraviti funkcijo `combine : 'a list -> 'a operator list -> 'a term option`, ki sprejme seznam vrednosti in seznam operatorjev ter vrne končen izraz, če je to mogoče, ali `None`.

3. naloga

France 9. februarja zjutraj prazni dolgo mizo, za katero je prejšnji večer praznovalo N gostov (vsi v eni sami vrsti). Vsak izmed njih je za sabo pustil nekaj (nenegativnih) enot hrane in pijače. Ker je Francetu med pospravljanjem dolgčas, se je odločil, da se bo namesto pospravljanja raje igral igrico.

Pospravljanje začne skrajno levo s praznim vrčem in se premika v desno. Ko pride na posamezno mesto, si vso pijačo, ki je ostala tam, pretoči v svoj kozarec in se vrne nazaj (v levo) za toliko mest, kot je gost pustil hrane. Ob vrnitvi na desno si ne natoči dodatne pijače, niti ne ponovi vračanja. Če bi se slučajno moral premaknit preko levega roba mize, potem se bo ob levem robu mize tudi ustavil. Ker je od prejšnjega dne še hudo utrujen, ob vsakem premiku (tako naprej kot nazaj) polije toliko enot pijače kot je le-ta dolg. Pri tem mora paziti, da ima v kozarcu vedno vsaj nekaj pijače. Dovoljeno je na primer, da s 3 enotami v kozarcu naredi premik dolg 3 enote, 4 pa je že preveč. Pri tem mora upoštevati tudi možne premike zaradi hrane in dodajanje pijače na posameznem mestu. S štirimi enotami se lahko premakne na mesto ki ima dve enoti hrane in je oddaljeno tri mesta, le, če ima to mest vsaj eno enoto pijače (drugače bi ob premiku nazaj ostal s praznim kozarcem). Zanima ga najmanj koliko mest bo obiskal, preden mu bo uspelo priti čez desni konec mize (z nenegativno količino pijače)? Pri tem ga budno opazuje celotna zbrana družba in skrbi, da se stanje na mizi ne spremeni (če bi dvakrat obiskal isto mesto, se je pijača tam vrnila na enako stanje, kot je bila na začetku).

Napišite funkcijo `pospravi`, ki sprejme seznam parov celih števil. Prva komponenta para pove količino hrane, druga pa pijače na posameznem mestu in vrne celo število, najmanjše število premikov, ki jih mora France opraviti, da se prebije do konca mize. Vaša rešitev lahko predpostavi, da bo to vedno mogoče v končno mnogo korakih.

Na spodnji mizi je najbolje, da najboljša pot, ki začne na indeksu 0 (si natoči dve enoti pijače), od koder se premakne na indeks 2 (kjer ima prazen kozarec) in se vrne nazaj na začetek, kjer ima sedaj 3 enote pijače. V drugo se spet premakne na indeks 2, in vrne nazaj na začetek s 4 enotami pijače. V tretje pa se premakne na indeks 1, in se vrne nazaj na začetek s 5 enotami pijače, s čimer ima dovolj zaloge da lahko s četrtim premikom doseže konec mize. (Alternativno bi lahko v tretjem premiku šel na indeks 2 in še vedno v enakem številu premikov dosegel konec mize).

```
[(1, 2), (1, 3), (3, 5), (5, 0), (5, 1)]
```