

Ime in priimek

--	--	--	--	--	--	--	--

Vpisna številka

1	
2	
3	
$\Sigma$	

## NAVODILA

- **Ne odpirajte te pole**, dokler ne dobite dovoljenja.
- **Preden začnete reševati test:**
  - Vpišite svoje podatke na testno polo z velikimi tiskanimi črkami.
  - Na vidno mesto položite osebni dokument s sliko in študentsko izkaznico.
  - Preverite, da imate mobilni telefon izklopljen in spravljen v torbi.
  - Prijavite se na spletno učilnico, kamor boste oddajali nekatere odgovore.
- Dovoljeni pripomočki: pisalo, brisalo, USB ključ, in poljubno pisno gradivo.
- Rešitve vpisujte v polo ali jih oddajte preko spletne učilnice. Pri odgovorih, ki ste jih oddali preko spletne učilnice, na izpitno nalogo napišite "glej spletno učilnico – datoteka <ime\_datoteke>".
- Če kaj potrebujete, prosite asistenta, ne sosedov.
- **Med izpitom ne zapuščajte svojega mesta** brez dovoljenja.
- Testna pola vam bo odvzeta **brez nadaljnjih opozoril**, če:
  - komunicirate s komerkoli, razen z asistentom,
  - komu podate kak predmet ali list papirja,
  - odrinete svoje gradivo, da ga lahko vidi kdo drug,
  - na kak drug način prepisujete ali pomagate komu prepisovati,
  - imate na vidnem mestu mobilni telefon ali druge elektronske naprave.
- **Ob koncu izpita:**
  - Ko asistent razglasi konec izpita, **takoj** nehajte in zaprite testno polo.
  - **Ne vstajajte**, ampak počakajte, da asistent pobere **vse** testne pole.
  - **Testno polo morate nujno oddati.**
- Čas pisanja je 120 minut. Na vidnem mestu je zapisano, do kdaj imate čas.
- Predvideni ocenjevalni kriterij:
  1.  $\geq 90$  točk, ocena 10
  2.  $\geq 80$  točk, ocena 9
  3.  $\geq 70$  točk, ocena 8
  4.  $\geq 60$  točk, ocena 7
  5.  $\geq 50$  točk, ocena 6

Veliko uspeha!

## 1. naloga (35 točk)

Klemen je za elbonijsko vlado v prologu implementiral program, ki izračuna vrednost elbonijskega aritmetičnega izraza. Sestavil je predikat `eval(E,N)`, ki velja, kadar ima aritmetični izraz `E` vrednost `N`.

```
:- use_module(library(clpfd)).

eval([N], N).

eval([+|E], A) :-
    append(E1, E2, E),
    eval(E1, A1),
    eval(E2, A2),
    A #= A1 + A2.

eval([-|E], A) :-
    append(E1, E2, E),
    eval(E1, A1),
    eval(E2, A2),
    A #= A1 - A2.

eval([*|E], A) :-
    append(E1, E2, E),
    eval(E1, A1),
    eval(E2, A2),
    A #= A1 * A2.
```

V prologu izraz zapišemo kot seznam operatorjev in števil, na primer izraz `+ - 3 1 5` zapišemo kot seznam `[+, -, 3, 1, 5]`.

**a) (7 točk)** Zapišite poizvedbo, s katero izračunate vrednost izraza `* + 1 2 - + 10 10 6`. Kakšen je odgovor?

**b) (7 točk)** Klemen je obiskal elbonijske študente računalništva in jim pokazal zgornji program. Vsakomur, ki bi znal iz programa razbrati slovnico aritmetičnih izrazov, je obljubil 0,00000000007 elbonijskih izpitnih točk, kar je ekvivalentno našim 7 točkam. Zapišite slovnico še vi v obliki BNF:

c) (7 točk) V programskem jeziku s podtipi velja  $\text{int} \leq \text{float}$ . Dana sta tipa zapisov

$$\begin{aligned}\rho &= \{\text{foo} : \text{float} \rightarrow \text{int}, \text{bar} : \text{int}, \text{baz} : \{\text{a} : \text{int}\}\}, \\ \tau &= \{\text{foo} : \text{float} \rightarrow \text{float}, \text{baz} : \{\}\}.\end{aligned}$$

Zapišite tip zapisa  $\sigma$ , da bo veljalo  $\rho \leq \sigma \leq \tau$  ter hkrati  $\sigma \not\leq \rho$  in  $\tau \not\leq \sigma$ . Pri tem uporabimo podtype zapisov po širini in globini.

d) (7 točk) Podatkovna struktura *2-3-drevo* je definirana induktivno:

- `Empty` je 2-3-drevo,
- če sta  $t$  in  $u$  2-3-drevesi in  $n \in \mathbb{Z}$ , je `Two`( $t, n, u$ ) 2-3-drevo,
- če so  $t, u$  in  $v$  2-3-drevesa in  $m, n \in \mathbb{Z}$ , je `Three`( $t, m, u, n, v$ ) 2-3-drevo.

Zapišite definicijo podatkovnega tipa 2-3-dreves v OCamlu ali Haskellu.

e) (7 točk) V  $\lambda$ -računu definirajte izraze `triple`, `fst`, `snd` in `thd`, da velja:

$$\text{fst}(\text{triple } X Y Z) = X, \quad \text{snd}(\text{triple } X Y Z) = Y, \quad \text{thd}(\text{triple } X Y Z) = Z.$$

## 2. naloga (30 točk)

Dokažite *popolno* pravilnost spodnjega programa. *Pišite čitljivo!*

```
[ y > 0 ]
x := 0 ;
c := 0 ;
while y > c do
  x := x + 1 ;
  c := x * x * x
end
if y < c then
  x := 0 ;
  y := 0
else
  skip
end
[ x =  $\sqrt[3]{y}$  ]
```

### 3. naloga (35 točk)

To nalogo lahko rešujete v OCamlu ali Haskellu.

**a) (10 točk)** *Vipavsko zaporedje*  $0, 1, -2, 5, -12, 29, -70, 169, \dots$  je definirano z rekurzivnim predpisom

$$x_0 = 0, \quad x_1 = 1, \quad x_n = x_{n-2} - 2 \cdot x_{n-1} \quad \text{za } n \geq 2.$$

Sestavite funkcijo `vipavsko : int -> int`, ki izračuna  $n$ -ti člen Vipavskega zaporedja. Primer uporabe:

```
# List.map vipavsko [0; 1; 2; 3; 4; 5; 6; 7] ;;  
- : int list = [0; 1; -2; 5; -12; 29; -70; 169]
```

Za vse točke mora funkcija  $n$ -ti člen izračunati v času  $O(n)$ , na primer `vipavsko 31` takoj vrne odgovor 259717522849. *Opomba:* to podnalogo lahko rešite kot poseben primer podnaloge (c).

Sedaj posplošimo pojem Vipavskega zaporedja. *Primorsko zaporedje*  $x_0, x_1, x_2, \dots$  je določeno s številoma  $a, b \in \mathbb{Z}$  in preslikavo  $f : \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}$  ter definirano takole:

$$x_0 = a, \quad x_1 = b, \quad x_n = f \, x_{n-2} \, x_{n-1} \quad \text{za } n \geq 2.$$

Primer: Vipavsko zaporedje dobimo pri  $a = 0$ ,  $b = 1$  in  $f \, m \, n = m - 2 \cdot n$ .

**b) (5 točk)** Zapišite  $a, b$  in  $f$ , da bo veljalo  $x_n = 2^n$ :

c) (20 točk) Sestavite funkcijo

```
zaporedje : int -> int -> (int -> int -> int) -> int -> int
```

ki sprejme  $a$ ,  $b$ ,  $f$  in  $n$  ter izračuna  $x_n$ . Primer uporabe:

```
# zaporedje 0 1 ( + ) 10 ;;  
- : int = 55
```

Za vse točke mora vaša rešitev izračunati  $n$ -ti člen zaporedja v času  $O(n)$ .