

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

Vpisna številka

1	
2	
3	
Σ	

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 pomagате 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. ≥ 90 točk, ocena 10
 2. ≥ 80 točk, ocena 9
 3. ≥ 70 točk, ocena 8
 4. ≥ 60 točk, ocena 7
 5. ≥ 50 točk, ocena 6

Veliko uspeha!

1. naloga (35 točk)

a) (7 točk) Elbonijsko ministrstvo za digitalno preobrazbo je objavilo slovnična pravila v BNF obliki za elbonijsko različico λ -računa, ki podpira nabore in vzorce:

$$\begin{aligned} \langle \text{spremenljivka} \rangle &::= [\text{a-z}]^+ \\ \langle \text{vzorec} \rangle &::= () \mid (\langle \text{spremenljivka} \rangle , \langle \text{vzorec} \rangle) \\ \langle \text{nabor} \rangle &::= () \mid (\langle \text{izraz} \rangle , \langle \text{nabor} \rangle) \\ \langle \text{izraz} \rangle &::= \langle \text{spremenljivka} \rangle \mid \langle \text{nabor} \rangle \mid \text{fun } \langle \text{vzorec} \rangle \Rightarrow \langle \text{izraz} \rangle \mid \langle \text{izraz} \rangle \langle \text{izraz} \rangle \mid (\langle \text{izraz} \rangle) \end{aligned}$$

Za vsakega od izrazov ugotovite, ali ustreza zgoraj podanim slovničnim pravilom:

Izraz	ustreza	ne ustreza
$(\text{a}, (\text{b}, ())) \text{ fun } (\text{x}, ()) \Rightarrow \text{x y}$	✓	
(a, b)		✓
$(\text{a}, ()) \text{ b})$		
$(\text{fun } () \Rightarrow \text{pes}, \text{fun } (\text{krava}, ()) \Rightarrow \text{mu})$		
$\text{fun } (\text{a}, (\text{b}, ())) \Rightarrow \text{a b}$		
n a r o b e		
$\text{fun } ((\text{a}, ()), (\text{b}, ())) \Rightarrow \text{a b}$		
$\text{fun a} \Rightarrow (\text{fun b} \Rightarrow \text{a b})$		
$\text{fun } () \Rightarrow () (\text{fun } () \Rightarrow ())$		

b) (7 točk) V prologu sestavite predikat `fib(L)`, ki velja natanko tedaj, ko je `L` seznam Fibonaccijevih števil v padajočem vrstnem redu:

```
?- fib([8,5,3,2,1,1,0]).
true.
?- fib([8,5,3,2,1,0]).
false.
?- fib([]).
true.
?- length(L,9), fib(L).
L = [21, 13, 8, 5, 3, 2, 1, 1, 0].
```

c) (7 točk) V funkcijskem programskem jeziku z zapisi in podtipi po širini in globini velja `int ≤ float`. Dani so naslednji tipi zapisov:

$$\begin{aligned} \rho &= \{a : (\text{float} \rightarrow \text{float}) \rightarrow \text{int}, b : \{c : \text{float}\}\} \\ \sigma &= \{a : (\text{int} \rightarrow \text{float}) \rightarrow \text{int}, c : \{b : \text{float}\}\} \\ \tau &= \{a : (\text{int} \rightarrow \text{float}) \rightarrow \text{int}, b : \{c : \text{int}\}\} \\ \psi &= \{b : \{\}\} \end{aligned}$$

Ugotovite, katera rezmerja veljajo:

	velja	ne velja
$\rho \leq \sigma$		
$\rho \leq \tau$		
$\tau \leq \rho$		
$\sigma \leq \rho$		
$\rho \leq \psi$		
$\sigma \leq \psi$		
$\tau \leq \psi$		

d) (7 točk) Standardna knjižnica za OCaml vsebuje funkciji z glavnima tipoma:

`List.fold_left : ($\alpha \rightarrow \beta \rightarrow \alpha$) $\rightarrow \alpha \rightarrow \beta$ list $\rightarrow \alpha$`
`List.append : γ list $\rightarrow \gamma$ list $\rightarrow \gamma$ list`

Izračunajte *glavni* tip izraza

`fun x \rightarrow List.fold_left List.append [[]] x`

e) (7 točk) V λ -računu predstavimo elemente množice permutacij reda 3 takole:

$p_{123} := \lambda f\ x\ y\ z. f\ x\ y\ z$
 $p_{132} := \lambda f\ x\ y\ z. f\ x\ z\ y$
 $p_{213} := \lambda f\ x\ y\ z. f\ y\ x\ z$
 $p_{231} := \lambda f\ x\ y\ z. f\ y\ z\ x$
 $p_{312} := \lambda f\ x\ y\ z. f\ z\ x\ y$
 $p_{321} := \lambda f\ x\ y\ z. f\ z\ y\ x$

Izračunajte normalno obliko (vrednost) izraza

$(\lambda p\ q\ f. p\ (q\ f))\ p_{132}\ p_{231}$

2. naloga (40 točk)

a) (20 točk) Fibonaccijevo zaporedje F_0, F_1, F_2, \dots je podano z rekurzivno definicijo

$$F_0 = 0, \qquad F_1 = 1, \qquad F_{i+2} = F_{i+1} + F_i.$$

Dokažite *delno* pravilnost programa, kjer je $n \in \mathbb{N}$:

```
{ true }  
x := 1 ;  
y := 2 ;  
i := 3 ;  
while x < n do  
  t := y ;  
  y := 1 + x + y ;  
  x := t ;  
  i := i + 1  
done ;  
x := x + 1  
{ x = Fi }
```

b) (20 točk) Dokažite še *polno* pravilnost programa, pri čemer smete brez dokaza upoštevati dejstvo, da za vse $i \geq 2$ velja $F_i < F_{i+1}$.

Namig. Spremenljivka n je konstanta.

(Prostor za reševanje)

3. naloga (30 točk)

Nalogo rešujte v programskem jeziku Haskell ali OCaml.

Vrtnično drevo (angl. *rose tree*) je podatkovna struktura, s katero predstavimo drevesa s poljubnim številom poddreves:

```
data Rose a = Thorn a | Rose [Rose a]
```

Z besedami: vrtnično drevo je bodisi trn, opremljen s podatkom tipa a , bodisi vozlišče s seznamom vrtničnih poddreves. Poznamo tudi običajna dvojiška drevesa:

```
data Tree a = Empty | Leaf a | Node (Tree a) (Tree a)
```

Z besedami: dvojiško drevo je bodisi prazno, bodisi list, opremljen s podatkom tipa a bodisi vozlišče z dvema dvojiškima poddrevesoma.

Vrtnično drevo $\text{Rose}[r_1, r_2, \dots, r_n]$ pretvorimo v dvojiško drevo tako, da gnezdimo dvojiška poddrevesa:

$$\text{Node } r'_1 (\text{Node } r'_2 (\dots \text{Node } r'_n \text{ Empty})),$$

kjer so r'_1, \dots, r'_n v dvojiška drevesa pretvorjena vrtnična poddrevesa r_1, \dots, r_n . Sami ugotovite, kako predstavimo $\text{Thorn } x$ in kako $\text{Rose}[]$.

a) (15 točk) Sestavite funkcijo $\text{rose2tree} : \text{Rose } a \rightarrow \text{Tree } a$, ki pretvori vrtnično drevo v dvojiško drevo. Primer uporabe:

```
> rose2tree (Rose [Thorn 'a', Thorn 'b', Thorn 'c'])
Node (Leaf 'a') (Node (Leaf 'b') (Node (Leaf 'c') Empty))
> rose2tree (Rose [Thorn 42, Rose [Thorn 23, Thorn 666, Rose []], Rose []])
Node (Leaf 42) (Node (Node (Leaf 23) (Node (Leaf 666) (Node Empty Empty))) (Node Empty Empty))
```

b) (15 točk) Sestavite funkcijo $\text{tree2rose} : \text{Tree } a \rightarrow \text{Rose } a$, ki dvojiško drevo pretvori nazaj v vrtnično drevo. Se pravi, da mora za vsako vrtnično drevo t veljati

$$\text{tree2rose} (\text{rose2tree } t) = t$$

Vseeno je, kaj funkcija naredi z dvojiškim drevesom, ki **ne predstavlja** vrtničnega drevesa. Primer uporabe:

```
> tree2rose (Node (Leaf 'a') (Node (Leaf 'b') (Node (Leaf 'c') Empty))))
Rose [Thorn 'a',Thorn 'b',Thorn 'c']
> tree2rose (Node (Node (Leaf 42) Empty) Empty)
Rose [Rose [Thorn 42]]
> tree2rose (Node Empty (Leaf 42))
Rose [Rose []]** Exception: not a rose tree
```

OCaml verzija:

```
type 'a rose = Thorn of 'a | Rose of 'a rose list
type 'a tree = Empty | Leaf of 'a | Node of 'a tree * 'a tree

# rose2tree (Rose [Thorn "a"; Thorn "b"; Thorn "c"]);;
- : string tree = Node (Leaf "a", Node (Leaf "b", Node (Leaf "c", Empty)))
# rose2tree (Rose [Thorn 42; Rose [Thorn 23; Thorn 666; Rose []]; Rose []]);;
- : int tree =
Node (Leaf 42,
  Node (Node (Leaf 23, Node (Leaf 666, Node (Empty, Empty))),
    Node (Empty, Empty)))

# tree2rose (Node (Leaf "a", Node (Leaf "b", Node (Leaf "c", Empty)))));;
- : string rose = Rose [Thorn "a"; Thorn "b"; Thorn "c"]
# tree2rose (Node (Node (Leaf 42, Empty), Empty));;
- : int rose = Rose [Rose [Thorn 42]]
# tree2rose (Node (Empty, Leaf 42));;
Exception: Failure "not a rose tree".
```

(Prostor za reševanje)