

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.
- Dovoljeni pripomočki: pisalo, brisalo, in poljubno pisno gradivo.
- Vse rešitve vpisujte v polo.
- 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.  $\geq 90$  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)

**a) (7 točk)** Ker Elbonija doživlja globoko ekonomsko krizo, na vsakem koraku varčujejo. Direktorat za aritmetiko je zato prepovedal nepotrebno uporabo podvojenih oklepajev, se pravi, da izraz ne sme vseovati podizrazov oblike  $((\dots))$ . Stara slovnica aritmetičnih izrazov dvojne oklepaje dopušča:

$$\begin{aligned}\langle \text{izraz} \rangle &::= \langle \text{multiplikativni} \rangle \mid \langle \text{izraz} \rangle + \langle \text{multiplikativni} \rangle \\ \langle \text{multiplikativni} \rangle &::= \langle \text{osnovni} \rangle \mid \langle \text{multiplikativni} \rangle \times \langle \text{osnovni} \rangle \\ \langle \text{osnovni} \rangle &::= \langle \text{število} \rangle \mid (\langle \text{izraz} \rangle) \\ \langle \text{število} \rangle &::= [0-9]^+\end{aligned}$$

Popravite jo tako, da bodo podvojeni oklepaji neveljavni. Na primer, izraz  $3 + (4 \times (5))$  je dovoljen, izraz  $3 + (4 \times ((5)))$  ni dovoljen zaradi  $((5))$  in izraz  $((3 + 4)) \times 5$  ni dovoljen zaradi  $((3 + 4))$ .

**b) (7 točk)** V logiki včasih poleg *neresnice*  $F$  in *resnice*  $T$  uporabljamo še vrednost *mogoče*  $M$ . Temu primerno razširimo tudi logične veznike, tako da na primer velja  $F \wedge p = F$  in  $T \vee p = T$ , ne glede na vrednost  $p$ . Natančneje, razširjena logična disjunkcija  $\vee$  je definirana takole:

$$p \vee q = \begin{cases} T & \text{če } p = T \text{ ali } q = T, \\ F & \text{če } p = F \text{ in } q = F, \\ M & \text{sicer.} \end{cases}$$

Klemen je predstavil razširjene resničnostne vrednosti v  $\lambda$ -računu z izrazi

$$T := \lambda x y z . x, \quad M := \lambda x y z . y, \quad F := \lambda x y z . z.$$

Zapišite izraz  $OR$ , ki izračuna razširjeni logični veznik  $\vee$ . Na primer, veljati mora

$$OR\ F\ q = q, \quad OR\ M\ F = M, \quad OR\ T\ F = T.$$

**c) (7 točk)** Izpeljite *glavni tip* OCaml funkcije

```
let twist f = fun (x, y) -> f (y, x)
```

Odgovor: \_\_\_\_\_

**d) (7 točk)** Sestavite program  $P$ , ki ustreza specifikaciji in dokažite njegovo pravilnost:

$$\begin{array}{c} \{ x > 0 \wedge y > 0 \} \\ P \\ \{ x > y \vee y > x \} \end{array}$$

**e) (7 točk)** Sestavite program  $Q$ , ki ustreza specifikaciji in dokažite njegovo pravilnost:

$$\begin{array}{c} \{ x > 0 \wedge y > 0 \} \\ Q \\ \{ x > y \wedge y > x \} \end{array}$$

## 2. naloga (35 točk)

Andrej je v OCamlu programiral različico klasične igrice Minesweeper. Igra poteka na minskem polju, v katerem položaje min in igralca predstavimo s celoštevilskimi koordinatami  $(x, y)$ . Igralec mora z zaporedjem korakov "dol", "gor", "levo" in "desno" prispeti od danega začetnega položaja do končnega, ne da bi stopil na mino. Korake predstavimo s podatkovnim tipom

```
type korak = Dol | Gor | Levo | Desno
```

**a) (15)** Sestavite funkcijo `premik : int*int -> korak -> int*int`, ki sprejme trenutni položaj in korak ter vrne naslednji položaj. Primeri:

```
# premik (0,3) Dol ;;
- : int * int = (0, 2)
# premik (0,3) Levo ;;
- : int * int = (-1, 3)
```

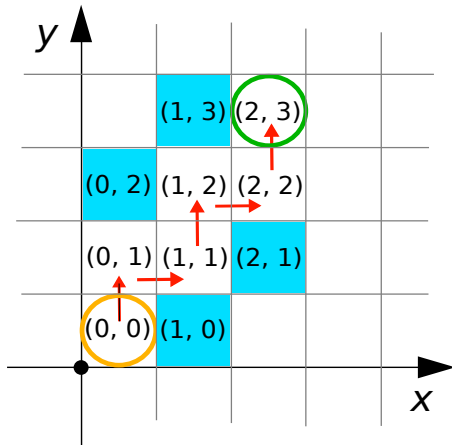
**b) (20)** Sestavite funkcijo

```
varna_pot : (int*int) list -> int*int -> int*int -> korak list -> bool
```

ki sprejme seznam položajev min, začetni položaj, končni položaj in seznam korakov. Funkcija vrne `true`, če dani seznam korakov vodi od začetnega do končnega položaja po poti, ki ne vsebuje mine. Primeri:

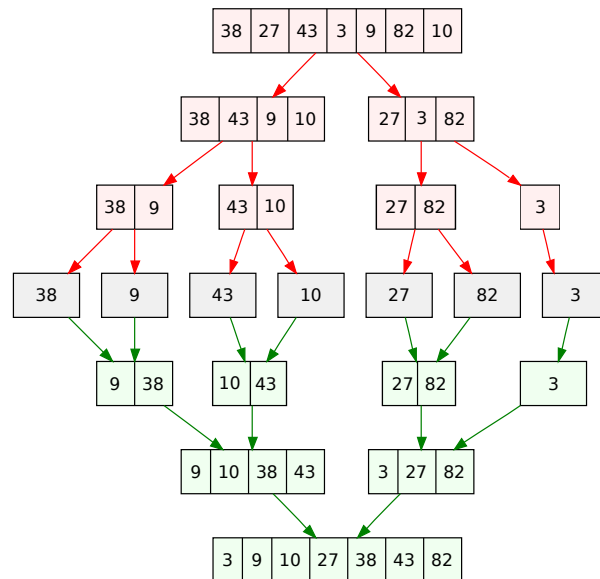
```
# varna_pot [] (0,0) (2,3) [Gor; Desno; Gor; Desno; Gor] ;;
- : bool = true
# varna_pot [(1,1)] (0,0) (2,3) [Gor; Desno; Gor; Desno; Gor] ;;
- : bool = false
# varna_pot [(0,2); (1,0); (1,3); (2,1)] (0,0) (2,3)
      [Gor; Desno; Gor; Desno; Gor] ;;
- : bool = true
```

Za vse točke naj bo funkcija *repno rekurzivna*.



### 3. naloga (30 točk)

V Prologu bomo sestavili program za urejanje seznamov z zlivanjem (angl. *merge sort*). To je postopek tipa "deli & vladaj", ki neurejeni seznam razdeli na dva enako dolga seznama, ju rekurzivno uredi in nato zlije v urejen seznam, kot to prikazuje spodnji primer. Pozor: običajno seznam razdelimo na pol, tu pa ga razdelimo na elemente na lihih in sodih mestih.



Najprej zapišimo glavni predikat `uredi(L, S)`, ki velja, kadar je `S` po velikosti urejen seznam `L`:

```

uredi([], []).
uredi([X], [X]).
uredi([X1,X2|Xs], LS) :-
    razdeli([X1,X2|Xs], Lihi, Sodi),
    uredi(Lihi, LihiS),
    uredi(Sodi, SodiS),
    zlij(LihiS, SodiS, LS).

```

Sedaj je treba sestaviti še predikata `razdeli` in `zlij`.

**a) (10 točk)** Sestavite predikat `razdeli(Xs, Ys, Zs)`, ki velja, kadar seznam `Xs` radelimo na seznama `Ys` in `Zs` tako, da so v `Ys` elementi iz lihih in v `Zs` elementi iz sodih položajev seznama `Xs`. Primeri:

```

?- razdeli([42], Ys, Zs).
Ys = [42], Zs = [].

?- razdeli([1,3,5,7,9,42], Ys, Zs).
Ys = [1, 5, 9], Zs = [3, 7, 42].

```

`razdeli(_____, _____, _____) .`  
`razdeli(_____, _____, _____) .`

`razdeli(_____, _____, _____) :-`  
`razdeli(_____, _____, _____) .`

**b) (20 točk)** Sestavite predikat `zlij(Xs, Ys, Zs)`, ki velja kadar je izpolnjen naslednji pogoj: če sta  $X_s$  in  $Y_s$  urejena seznama, je  $Z_s$  urejen seznam elementov iz  $X_s$  in  $Y_s$ . Primeri:

```
?- zlij([3,5,6,10,14], [1,2,5,6], Zs).  
Zs = [1, 2, 3, 5, 5, 6, 6, 10, 14] ;  
false.
```

```
?- zlij([], [1], Zs).  
Zs = [1].
```

Rešitev: