

Ime in priimek _____

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

Vpisna številka

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.
 - Prjavite se na spletno učilnico, kamor boste oddajali odgovore.
- Dovoljeni pripomočki: pisalo, brisalo, USB ključ in poljubno pisno gradivo.
- Vse rešitve vpisujte v kviz na spletni učilnici.
- 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 tabli je zapisano, do kdaj imate čas.
- Predvideni ocenjevalni kriterij:
 - ≥ 90 točk, ocena 10
 - ≥ 80 točk, ocena 9
 - ≥ 70 točk, ocena 8
 - ≥ 60 točk, ocena 7
 - ≥ 50 točk, ocena 6

Veliko uspeha!

1. naloga (30 točk)

a) (6 točk) Elbonija uporablja naslednjo sintakso za zapis aritmetičnih izrazov:

$$\langle \text{izraz} \rangle ::= \langle \text{številka} \rangle \mid \ominus \langle \text{izraz} \rangle \mid \langle \text{izraz} \rangle \oplus \langle \text{izraz} \rangle \mid \langle \text{izraz} \rangle \otimes \langle \text{izraz} \rangle \\ \langle \text{številka} \rangle ::= [0-9]^+$$

Simboli \ominus , \oplus in \otimes označujejo nasprotno vrednost, seštevanje in množenje. Elbonijsko razumevanje aritmetike je pomanjkljivo, zato nimajo nikakršnih dogovorov o prioriteti in asociativnosti operacij, prav tako pa ne poznajo oklepajev. Na primer, elbonijski izraz $2 \otimes 3 \oplus 4$ bi po naše lahko razumeli kot $(2 \cdot 3) + 4$ ali kot $2 \cdot (3 + 4)$.

Katere so možne vrednosti izraza $\ominus 2 \oplus 3 \otimes 4$?

- (i) 4 in 10
- (ii) -20 , -14 , 4 in 10
- (iii) -20 , -14 in 10
- (iv) 10

b) (6 točk) V okolju $[n \mapsto 10, i \mapsto 0, v \mapsto 0]$ evaluiramo program

```
v := 0 ;
while i < n do
  v := v + i ;
  i := i + 1
done
```

Kakšno bo okolje, ko bo program končal?

- (i) $[n \mapsto 10, i \mapsto 9, v \mapsto 45]$
- (ii) $[n \mapsto 10, i \mapsto 10, v \mapsto 45]$
- (iii) $[n \mapsto 10, i \mapsto 10, v \mapsto 55]$
- (iv) $[n \mapsto 10, i \mapsto 11, v \mapsto 55]$

c) (6 točk) Definiramo λ -izraze

$$I = \lambda x . x, \quad K = \lambda x y . x, \quad L = \lambda x y . y.$$

Katere od naslednjih enakosti veljajo? (Napačno izbrani odgovori štejejo -2 točki.)

- (i) $K I L = L L L L$
- (ii) $K I L L = L L L$
- (iii) $K I L L L = L L$
- (iv) $K I L L L L = L$

d) (6 točk) Kateri od naslednjih prolog programov je ekvivalenten formuli $(a \vee b) \Rightarrow (c \wedge d)$?

(i) $c :- a, b.$
 $d :- a, b.$

(ii) $c :- a; b.$
 $d :- a; b.$

(iii) $a :- c; d.$
 $b :- c; d.$

(iv) $a :- c, d.$
 $b :- c, d.$

e) (6 točk) V turistični agenciji *Bratko Travels* uporabljajo prolog. Letalske povezave med letališči predstavijo z relacijo `polet(X, Y, T)`, ki pomeni, da obstaja polet met letališčema x in y , ki traja T ur. Primer podatkov, ki jih hranijo:

```
polet(ljubljana, frankfurt, 1).  
polet(ljubljana, amsterdam, 2).  
polet(ljubljana, london, 2).  
polet(frankfurt, ljubljana, 1).  
polet(frankfurt, london, 2).  
polet(frankfurt, amsterdam, 1).  
polet(frankfurt, newyork, 9).  
polet(amsterdam, ljubljana, 2).  
polet(amsterdam, frankfurt, 1).  
polet(amsterdam, newyork, 6).  
polet(london, ljubljana, 2).  
polet(london, frankfurt, 2).  
polet(london, newyork, 6).  
polet(newyork, frankfurt, 9).  
polet(newyork, amsterdam, 6).  
polet(newyork, london, 6).
```

Zapišite poizvedbo v prologu, s katero ugotovimo, ali je možno leteti od Ljubljane do New Yorka z natanko enim prestopanjem tako, da je skupni čas letenja manjši od 9 ur.

2. naloga (35 točk)

Dokažite *popolno* pravilnost programa, kjer je n pozitivno celo število:

```
[  $0 < n$  ]  
k := 1 ;  
while k * k < n do  
  k := k + 1  
done  
[  $(k - 1)^2 < n \leq k^2$  ]
```

Opomba: delna pravilnost je vredna 20 točk, dokaz zaustavitve programa pa 15 točk.

3. naloga (35 točk)

Obravnavajmo aritmetične izraze s konstantami, spremenljivkami in seštevanjem:

$$\begin{aligned}\langle \text{expression} \rangle &::= \langle \text{number} \rangle \mid \langle \text{variable} \rangle \mid \langle \text{expression} \rangle + \langle \text{expression} \rangle \\ \langle \text{number} \rangle &::= [0-9]^+ \\ \langle \text{variables} \rangle &::= [a-z]^+\end{aligned}$$

V OCamlu definiramo podatkovni tip, ki predstavlja abstraktno sintakso izrazov, in funkcijo

```
eval : (string * int) list -> expression -> int
```

ki v danem okolju evaluiira dani izraz:

```
type expression =
  | Number of int
  | Var of string
  | Plus of expression * expression

let rec eval env = function
  | Number k -> k
  | Var x -> List.assoc x env
  | Plus (e1, e2) -> eval env e1 + eval env e2
```

a) (5 točk) V OCamlu definirajte vrednost `izraz`, ki predstavlja izraz $3 + (x + 5)$, nato pa z uporabo funkcije `eval` izračunajte njegovo vrednost v okolju $[(x, 7), (y, 2)]$:

```
let izraz = ...
let vrednost = eval ...
```

b) (30 točk) Če upoštevamo, da je seštevanje asociativno in komutativno, lahko izraze optimiziramo tako, da vse celoštevilске konstante zberemo skupaj. Na primer:

$$\begin{aligned}(2 + 3) + 1 &= 6 \\ (x + 5) + (y + (x + 2)) &= 7 + x + y + x \\ 0 + x &= x\end{aligned}$$

Vrstni red spremenljivk smo ohranili, prav tako nismo združili dveh x v $2 \cdot x$, ker nimamo množenja. Kako asociiramo rezultat, ni pomembno.

Sestavite funkcijo

```
optimize : expression -> expression
```

ki sprejme izraz in ga optimizira, kot je prikazano zgoraj. Primeri uporabe:

```
# optimize (Plus (Number 2, Plus (Number 3, Number 1))) ;;
- : expression = Number 6
# optimize (Plus (Plus (Var "x", Number 5),
  Plus (Var "y", Plus (Var "x", Number 2)))) ;;
- : expression = Plus (Plus (Plus (Number 7, Var "x"), Var "y"), Var "x")
# optimize (Plus (Number 0, Var "x")) ;;
- : expression = Var "x"
```

Namig: profesor Bauer je nalogo rešil tako, da je definiral dve pomožni funkciji: prva iz danega izraza izračuna celoštevilsko konstanto in seznam spremenljivk, ki se pojavijo v njem; druga iz celoštevilске konstante in seznama spremenljivk sestavi izraz.