Principi programskih jezikov

2. izpit, 30. junij 2025

1	
2	
3	
Σ	

Ime in priimek

Vpisna številka

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 mobitel izklopljen in spravljen v torbi.
  - Prijavite se na spletno učilnico, kamor boste oddajali nekatere odgovore.
- Dovoljeni pripomočki: pisalo, brisalo, gradivo predčasno naloženo na spletni učilnici, in poljubno pisno gradivo.
- Rešitve vpisujte v polo ali jih oddajte preko spletne učilnice. Pri odgovorih, ki ste jih oddali na spletni učilnici, na poli zapišite *"odgovor je v datoteki* `<ime_datoteke>`*"*.
- Če kaj potrebujete, prosite asistenta in ne sosedov.
- **Med izpitom ne zapuščajte svojega mesta** brez dovoljenja.
- Testna pola vam bo odvzeta **brez nadaljnjih opozoril**, če:
  - komunicirate s komerkoli (ali LLM-i), 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 mobitel 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)

Elbonijska obveščevalna služba je ugotovila, da v sovražni državi Kneeboniji uporabljajo naslednjo slovnico za aritmetične izraze, pri čemer imata + in × enako prioriteto:

$$\langle\text{izraz}\rangle ::= \langle\text{število}\rangle \mid \langle\text{izraz}\rangle + \langle\text{izraz}\rangle \mid \langle\text{izraz}\rangle \times \langle\text{izraz}\rangle \mid (\ \langle\text{izraz}\rangle\ )$$
$$\langle\text{število}\rangle ::= [0\text{–}9]+$$

Kneebonijska slovnica je dvoumna: $1 + 2 \times 3$ lahko razčlenimo kot $(1 + 2) \times 3$ in kot $1 + (2 \times 3)$.

Elbonijci načrtujejo uničujoč kibernetski napad na Kneebonijo. Po elektronski pošti jim bodo poslali aritmetični izraz, ki ga je možno razčleniti na vsaj 1000 načinov in s tem preobremenili Kneebonijski Centralni Kalkulator. Če zapišete tak izraz, vam bodo v znak hvaležnosti podelili 7 točk.

b) (7 točk) V λ-računu predstavimo seznam $[e_1, e_2, \ldots, e_n]$ takole:

$$[e_1, e_2, \ldots, e_n] := \lambda\ f\ b\,.\ f\ e_1\ (f\ e_2\ (\ldots(f\ e_n\ b)\ldots)) \qquad (\text{če } n > 0)$$
$$[\,] := \lambda\ f\ b\,.\ b$$

Poiščite λ-izraz `cons`, ki stakne element in seznam, se pravi, da velja

$$\texttt{cons}\ d\ [e_1, \ldots, e_n] = [d, e_1, \ldots, e_n].$$

c) (7 točk) V programskem jeziku s parametričnim polimorfizmom definiramo rekurzivno funkcijo

```
let rec yes x = yes (print_endline "yes")
```

Zunanja funkcija `print_string` ima tip `string -> unit`. Izračunajte *glavni* tip izraza

```
yes ()
```

Za vse točke mora biti razviden postopek reševanja.

d) (7 točk) Ali sta v ukaznem programskem jeziku ukaza

`if` b `then (`A `;` C `;` Z`) else (`A `;` D `;` Z`) end`

in

A `; (if` b `then` C `else` D `end) ;` Z

ekvivalentna za vse logične izraze b in ukaze A, C, D in Z? Ekvivalenco bodisi dokažite bodisi podajte *konkreten* protiprimer.

e) (7 točk) V preprostem programskem jeziku z osnovnima tipoma `int` in `float` ter funkcijskimi tipi vpeljemo relacijo podtip $\leq$ s pravili (in nobenimi drugimi!)

$$\frac{}{\texttt{int} \leq \texttt{int}} \qquad \frac{}{\texttt{int} \leq \texttt{float}} \qquad \frac{}{\texttt{float} \leq \texttt{float}} \qquad \frac{\tau_2 \leq \tau_1 \qquad \sigma_1 \leq \sigma_2}{(\tau_1 \to \sigma_1) \leq (\tau_2 \to \sigma_2)}$$

V prologu definirajte dvomestno relacijo `sub(X,Y)`, ki predstavlja $\leq$, upoštevajoč dana pravila. Funkcijski tip $\sigma \to \tau$ v prologu predstavimo z izrazom $\texttt{arrow}(\sigma, \tau)$. Primer poizvedbe, s katero izračunamo vse podtipe tipa $(\texttt{int} \to \texttt{int}) \to (\texttt{float} \to \texttt{float})$:

```
?- sub(X, arrow(arrow(int, int), arrow(float, float))).
X = arrow(arrow(int, int), arrow(float, int)) ;
X = arrow(arrow(int, int), arrow(float, float)) ;
X = arrow(arrow(int, float), arrow(float, int)) ;
X = arrow(arrow(int, float), arrow(float, float)) ;
false.
```

2. naloga (35 točk)

Klemen je sestavil skrivnostni program S in dokazal, da zanj velja popolna pravilnost

$[\top]$
S
$[\mathtt{x} = \mathtt{n}]$

a) (25 točk) Dokažite *delno* pravilnost programa, v katerem se pojavi skrivnostni program:

```
{ n > 1 }
x := n ;
c := n ;
while x = n and not (c = 0) do
  S ;
  c := c - 1
done
{ c = 0 }
```

Vse spremenljivke zavzemajo celoštevilske vrednosti. Poskrbite, da bo v vaši rešitvi razvidna invarianta zanke.

b) (10 točk) Dokažite še popolno pravilnost, ali pa podajte primer programa S, ki zadošča Klemnovi popolni pravilnost, a za katerega popolna pravilnost programa iz podnaloge (a) ne velja.

3. naloga (40 točk)

To nalogo lahko rešujete v OCamlu ali Haskellu.

Definirajmo podatkovni tip Boolovih izrazov s spremenljivkami:

```
type izraz =
  | True
  | False
  | Var of string
  | Not of izraz
  | And of izraz * izraz
  | Or of izraz * izraz
```

Na primer, boolov izraz $x \wedge \neg(y \vee \bot)$ predstavimo z

```
And (Var "x", Not (Or (Var "y", False)))
```

Okolje je končna preslikava $x_1 \mapsto b_1, \ldots, x_n \mapsto b_n$, ki spremenljivkam priredi boolove vrednosti. Predstavimo ga z asociativnim seznamom, na primer okolje $x \mapsto \bot, y \mapsto \bot, z \mapsto \top$ predstavimo z `[("x", false); ("y", false); ("z", true)]`.

a) (15 točk) Sestavite funkcijo

```
eval : (string * bool) list -> izraz -> bool
```

ki sprejme okolje in izraz ter izračuna njegovo vrednost v danem okolju. Primeri uporabe:

```
# eval [] (Or (And (False, True), False)) ;;
- : bool = false
# eval ["x",false] (Not (Var "x")) ;;
- : bool = true
# eval [("x",false); ("y",false); ("z",true)]
       (And (Var "x", Not (Or (Var "y", False)))) ;;
- : bool = false
```

Predpostaviti smete, da okolje vsebuje vse spremenljivke, ki se pojavijo v izrazu.

b) (10 točk) Sestavite funkcijo `eval' : izraz -> bool option`, ki izračuna vrednost danega izraza, če ta ne vsebuje spremenljivk. Natančneje, `eval' e` vrne `None`, če se v izrazu `e` pojavi kaka spremenljivka, sicer vrne `Some b`, kjer je `b` vrednost izraza `e`. Primeri uporabe:

```
# eval' (Or (And (False, True), False)) ;;
- : bool option = Some false
# eval' (Not (Var "x")) ;;
- : bool option = None
# eval' (And (Var "x", Not (Or (Var "y", False)))) ;;
- : bool option = None
```

c) (15 točk) Sestavite funkcijo `optimize : izraz -> izraz`, ki izraz poenotavi tako, da izračuna vrednosti podizrazov brez spremenljivk, ter upošteva enačbe

$$p \wedge \top = p, \qquad p \wedge \bot = \bot, \qquad p \vee \bot = p, \qquad p \wedge \top = p$$
$$\top \wedge p = p, \qquad \bot \wedge p = \bot, \qquad \bot \vee p = p, \qquad \top \wedge p = p.$$

Primeri uporabe:

```
# optimize (And (True, Or (False, Var "x"))) ;;
- : izraz = Var "x"
# optimize (Or (Var "x", Not (Var "x"))) ;;
- : izraz = Or (Var "x", Not (Var "x"))
# optimize (And (Var "x", Not (Or (Var "y", False)))) ;;
- : izraz = And (Var "x", Not (Var "y"))
```