Principi programskih jezikov, 3. izpit
23. avgust 2019

								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.
  - 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 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 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 (40 točk)

a) (8 točk) V λ-računu definiramo funkciji

$$\mathsf{true} := \lambda xy\,.\,x \qquad \text{in} \qquad \mathsf{false} := \lambda xy\,.\,y.$$

Katera od naslednjih funkcij predstavlja operacijo xor, torej vrne true, če sta argumenta p in q različni boolovi vrednosti, in false, če sta enaki boolovi vrednosti:

1. $\lambda p\,q\,.\,p\,(q\,\mathsf{false}\,\mathsf{true})\,q$
2. $\lambda p\,q\,.\,p\,q\,p$
3. $\lambda p\,q\,.\,p\,q\,q$
4. $\lambda p\,q\,.\,(q\,\mathsf{false}\,\mathsf{true})\,p\,q$

b) (8 točk) Obravnavamo naslednje izjave o pravilnosti programa P:

(1) $\{n = 1\}\ P\ \{\mathsf{false}\}$

(2) $[n = 1]\ P\ [\mathsf{false}]$

(3) $\{n = 1\}\ P\ \{\mathsf{true}\}$

(4) $[n = 1]\ P\ [\mathsf{true}]$

ter naslednje razlage:

(A) Če je $n = 1$, se program P ustavi.

(B) Če je $n = 1$, se program P ne ustavi.

(C) $n \neq 1$.

(D) Izjava velja za vse P.

Vsaki od izjav priredite njeno razlago:

(1): _____ (2): _____ (3): _____ (4): _____

c) (8 točk) V OCamlu zapišite kako funkcijo, ki ima tip *natanko* `(('a -> 'a) -> 'b) -> 'b`.

d) (8 točk) V jeziku z zapisi uporabljamo podtipe v širino in globino. Relacijo podtip označimo z $\leq$ in definiramo tipa zapisov:

$$\mathsf{type}\ s = \{x : \{a : \mathtt{int}\}, y : \{b : \mathtt{int} \to \{\ \}\}\}$$
$$\mathsf{type}\ u = \{x : \{a : \mathtt{int}\}\}.$$

Zapišite tak tip t, da bo veljalo $s \leq t \leq u$, hkrati pa $t \neq s$ in $t \neq u$.

e) (8 točk) Usmerjen graf z vozlišči v prologu predstavimo s predikatom `povezava/2`, kjer `povezava(X,Y)` pomeni, da od vozlišča `X` do vozlišča `Y` poteka povezava.

Dan je naslednji graf z vozlišči a, b, c, d, e:

```
povezava(a, b).
povezava(b, c).
povezava(c, a).
povezava(c, d).
povezava(d, e).
povezava(e, c).
povezava(b, e).
```

Trikotnik je taka trojica vozlišč X, Y, Z, da je X povezan z Y, Y z Z in Z z X. V prologu definiramo trikotnik takole:

```
trikotnik(X,Y,Z) :-
  povezava(X, Y),
  povezava(Y, Z),
  povezava(Z, X).
```

Koliko trikotnikov najde v zgornjemm grafu poizvedba

```
?- trikotnik(X,Y,Z).
```

2. naloga (30 točk)

Elbonijci so se naveličali sintakse artimetičnih izrazov. Te dni jih bolj zanimajo *neprazna* dvojiška zaporedja ničel in enic. Ker so vraževerni, verjamejo, da so uročena vsa zaporedja, ki vsebujejo tri zaporedne ničle. Na primer, zaporedja 000, 100011, 10010000 so uročena, medtem ko zaporedja 1, 10, 00100, 00111 niso uročena.

Slovnična pravila za *vsa* neprazna zaporedja lahko predstavimo s slovnico:

$$\langle \text{zaporedje} \rangle ::= \texttt{0} \mid \texttt{1} \mid \texttt{0}\langle \text{zaporedje} \rangle \mid \texttt{1}\langle \text{zaporedje} \rangle$$

V svojih rešitvah smete privzeti slovnično pravilo $\langle$zaporedje$\rangle$.

a) (15 točk) Zapišite slovnična pravila za *uročena* neprazna zaporedja, se pravi taka, ki vsebujejo tri zaporedne ničle.

b) (15 točk) Zapišite slovnična pravila za *neuročena* neprazna zaporedja, se pravi taka, ki *ne* vsebujejo treh zaporednih ničel.

3. naloga (30 točk)

Timotej se ukvarja z optimizacijo preprostega zbirnika, ki ima samo dva ukaza:

1. ukaz `MOV` i, j prebere vrednost pomnilniške lokacije i in jo zapiše v pomnilniško lokacijo j,
2. ukaz `ADD` i, j, k prebere vrednosti pomnilniških lokacij i in j ter njuno vsoto zapiše v lokacijo k.

Pomnilniške lokacije naslavljamo z nenegativni celimi števili. Na primer, če je stanje pomnilnika $[1, 2, 5]$ in izvedemo program

`MOV` $1, 0$
`ADD` $2, 2, 1$
`ADD` $0, 1, 2$

dobimo pomnilnik $[2, 10, 12]$.

V OCamlu predstavimo ukaz z vrednostjo tipa

```
type instruction =
  | Mov of int * int
  | Add of int * int * int
```

in pomnilnik s seznamom celih števil.

a) (10 točk) Sestavite funkcijo

```
val writes_to : int -> instruction -> bool
```

kjer `writes_to k c` ugotovi, ali ukaz `c` zapiše vrednost v lokacijo `k`, se pravi, da je oblike `MOV` i, k ali `ADD` i, j, k.

b) (10 točk) Sestavite funkcijo

```
val reads_from : int -> instruction -> bool
```

kjer `reads_from k c` ugotovi, ali ukaz `c` prebere vrednost lokacije `k`, se pravi, da je oblike `MOV` k, j ali `ADD` i, k, j ali `ADD` k, i, j.

c) (10 točk) Timotej je ugotovil, da lahko zaporedje ukazov optimizira tako, da nekatere ukaze zbriše, ne da bi to vplivalo na učinek programa. Postopek optimizacije zaporedja ukazov $[c_1; c_2; \ldots; c_n]$ je naslednji:

1. Optimizramo rep zaporedja $[c_2; \ldots; c_n]$ in dobimo zaporedje $[c_2'; \ldots; c_m']$.
2. Denimo, da ukaz c_1 zapiše vrednost v lokacijo k. Tedaj:
   (a) če c_2' bere z lokacije k, potem ukaza c_1 ne smemo odstraniti s seznama, sicer
   (b) če c_2' piše na lokacijo k, potem smemo c_1 odstraniti, sicer
   (c) je c_2' neodvisen od lokacije k, zato preverimo iste pogoje za $[c_3'; \ldots; c_m']$.

Na primer, zgornji postopek optimizira ukaze na levi v ukaze na desni:

`MOV` $1, 2$		
`MOV` $0, 1$		`MOV` $1, 2$
`MOV` $4, 4$		`MOV` $4, 4$
`MOV` $1, 3$	$\longrightarrow$	`ADD` $0, 0, 3$
`ADD` $0, 0, 3$		`MOV` $2, 1$
`MOV` $2, 1$		

Sestavite funkcijo

```
val optimize : instruction list -> instruction list
```

ki optimizira seznam ukazov glede na zgornja pravila.