

Principi programskih jezikov

2. izpit, 20. junij 2024

Ime in priimek

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

Vpisna številka

1	
2	
3	
4	
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, gradivo predčasno naloženo na e-učilnici, 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 pomagate 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 (28 točk)

a) (7 točk) Za vsakega izmed parov naslednjih programov v OCamlu v praznih celicah z DA/NE označite, če sta ekvivalentna.

1. `fun x y -> x + y`

2. `fun y x -> x + y`

3. `fun x -> x + y`

4. `fun y -> x + y`

5. `fun z -> z + y`

6. `x + y`

7. `z + y`

	1	2	3	4	5	6	7
1	—						
2	—	—					
3	—	—	—				
4	—	—	—	—			
5	—	—	—	—	—		
6	—	—	—	—	—	—	
7	—	—	—	—	—	—	—

b) (7 točk) Izračunajte (na dolgo) *glavni* tip naslednjega izraza:

```
let rec f s1 s2 =  
  match (s1, s2) with  
  | (h1::t1, h2::t2) -> (h1, h2) :: f t1 t2  
  | _ -> []
```

c) (14 točk) V prologu definirajte predikat `calc(Env, Expr, Result)`, ki velja, ko je `Result` izračunana vrednost izraza `Expr` v okolju `Env`. Pri tem je izraz `Expr` lahko število, spremenljivka iz okolja ter vsota `E1+E2`, razlika `E1-E2` ali produkt izrazov `E1*E2`. Okolje `Env` je podano v obliki seznama z elementi oblike `Ime_spremenljivke-Vrednost_spremenljivke`. Pomagajte si s predikatom `number(N)`, ki velja, ko je `N` število.

Primeri.

```
?- calc([a-15, b-100], c, R).  
false.
```

```
?- calc([a-15, b-100], b, R).  
R = 100 ;  
false.
```

```
?- calc([a-15, b-100], b+a, R).  
R = 115 ;  
false.
```

```
?- calc([a-15], 100-a, R).  
R = 85 ;  
false.
```

```
?- calc([a-15, b-100], b*a, R).  
R = 1500 ;  
false.
```

```
?- calc([a-15, b-100], a-b*a, R).  
R = -1485 ;  
false.
```

```
?- calc([a-15, b-100], (a-b)*a, R).  
R = -1275 ;  
false.
```

```
?- calc([a-15, b-100], a*10-(101-b)*(-2), R).  
R = 152 ;  
false.
```

2. naloga (21 točk)

V OCaml-u smo definirali naslednja dva tipa:

```
type size = Small | Medium | Large
type 'a classes = { small : 'a list; medium : 'a list; large : 'a list }
```

Implementiraj funkcijo `sort : ('a -> size) -> 'a list -> 'a classes`, ki glede na funkcijo `'a -> size` razdeli elemente vhodnega seznama `'a list` v tri sezname v obliki zapisa `'a classes`. Vrstni red elementov ni pomemben. Za vse točke naj tvoja rešitev dela tudi za zelo dolge sezname ($> 10^8$ elementov).

Primeri.

```
# let f x = if x < 10 then Small else if x < 100 then Medium else Large;;
val f : int -> size = <fun>

# sort f [1003; 33; 4; 1; 20; 3; 550; 23; 88; 300];;
- : int classes =
{small = [3; 1; 4]; medium = [88; 23; 20; 33]; large = [300; 550; 1003]}

# sort f [1; 2; 3; 4; 5; 6; 7; 8; 9; 10];;
- : int classes =
{small = [9; 8; 7; 6; 5; 4; 3; 2; 1]; medium = [10]; large = []}

# sort f (List.init 100000000 (fun i -> i));; (* after 20s or more *)
- : int classes =
{small = [9; 8; 7; 6; 5; 4; 3; 2; 1; 0];
 medium =
  [99; 98; 97; 96; 95; 94; 93; 92; 91; 90; 89; 88; 87; 86; 85; 84; 83; 82; 81;
   80; 79; 78; 77; 76; 75; 74; 73; 72; 71; 70; 69; 68; 67; 66; 65; 64; 63; 62;
   61; 60; 59; 58; 57; 56; 55; 54; 53; 52; 51; 50; 49; 48; 47; 46; 45; 44; 43;
   42; 41; 40; 39; 38; 37; 36; 35; 34; 33; 32; 31; 30; 29; 28; 27; 26; 25; 24;
   23; 22; 21; 20; 19; 18; 17; 16; 15; 14; 13; 12; 11; 10];
 large =
  [99999999; 99999998; 99999997; 99999996; 99999995; 99999994; 99999993; ...
```

3. naloga (35 točk)

a) (25 točk) Dokazite *delno* pravilnost spodnjega programa. Zapišite, katere invariante ste uporabili.

```
{ N >= 0 }
i := -1 ;
j := -1 ;
x := N ;
while x >= 0 do
  i := i + 1 ;
  x := x - 100 ;
done
x := x + 100 ;
while x >= 0 do
  j := j + 1 ;
  x := x - 10
done
x := x + 10
{ N = 100i + 10j + x }
```

b) (10 točk) Dokazite popolno pravilnost programa.

4. naloga (21 točk)

a) (14 točk) V Prologu definirajte predikat `min_rest(L, Min, Rest)`, ki velja, ko je `Min` najmanjše število v seznamu `L`, seznam `Rest` pa seznam `L` brez elementa `Min`. Če imamo v seznamu podvojene elemente, odstranimo prvega z desne (glejte primere).

```
?- min_rest([4,2,5,1,6], Min, Rest).
Min = 1,
Rest = [4, 2, 5, 6] ;
false.

?- min_rest([], Min, Rest).
false.

?- min_rest([-1], Min, Rest).
Min = -1,
Rest = [] ;
false.

?- min_rest([1,3,-1,4,-1,6], Min, Rest).
Min = -1,
Rest = [1, 3, -1, 4, 6] ;
false.
```

b) (7 točk) V Prologu definirajte predikat `min2(L, Min1, Min2)`, ki velja, ko je `Min1` najmanjše število in `Min2` drugo najmanjše število v seznamu `L`. Primeri:

```
?- min2([], Min1, Min2).
false.

?- min2([-1], Min1, Min2).
false.

?- min2([1,-1], Min1, Min2).
Min1 = -1,
Min2 = 1 ;
false.

?- min2([4,2,5,1,6], Min1, Min2).
Min1 = 1,
Min2 = 2 ;
false.

?- min2([4,2,5,2,6], Min1, Min2).
Min1 = Min2, Min2 = 2 ;
false.
```