

Programiranje II: poskusni izpit

17. april 2024

Čas reševanja je 60 minut. Veliko uspeha!

1. naloga (10 točk)

Za vsakega izmed spodnjih programov prikažite vse spremembe sklada in kopice, če poženemo funkcijo main. Za vsako spremembo označite, po kateri vrstici v kodi se zgodi.

a)

```
1  fn f(a: i32, b: i32) -> i32 {
2      a * b
3  }
4  fn g(x: i32) -> i32 {
5      f(x, x + 1)
6  }
7  fn main() {
8      let m = 6;
9      let n = g(m);
10     println!("{n}")
11 }
```

b)

```
1  fn f(s: String) {
2      println!("{s}")
3  }
4  fn g(s: String) {
5      f(s)
6  }
7  fn main() {
8      let s2 = String::from("2");
9      let s1 = String::from("4");
10     if true {
11         println!("{s2}");
12     }
13     g(s1);
14 }
```

c)

```
1  fn f(s: &String) {
2      println!("{s}")
3  }
4  fn g(s: String) {
5      f(&s)
6  }
7  fn main() {
8      let s1 = String::from("4");
9      let s2 = String::from("2");
10     g(s1);
11     println!("{s2}");
12 }
```

2. naloga (10 točk)

Definirajmo tip množic `Set<T>`. Dopolnite signature spodnjih metod. Če v dani prostor ni treba dopisati ničesar, ga prečrtajte.

- a) `fn contains(_____ self, x: _____) _____`, ki preveri, ali dana množica vsebuje element `x`.
- b) `fn power_set(_____ self) _____`, ki vrne potenčno množico dane množice.
- c) `fn intersection(_____ self, other: _____) _____`, ki izračuna presek dveh množic.
- d) `fn add(_____ self, x: _____) _____`, ki v obstoječo množico doda element `x`.
- e) `fn into_iter(_____ self) _____`, ki iz množice naredi iterator po njenih elementih.

3. naloga (30 točk)

Za vsakega izmed spodnjih programov:

1. razložite, zakaj in s kakšnim namenom Rust program zavrne;
2. program popravite tako, da bo veljaven in bo učinkovito dosegel prvotni namen.

a)

```
fn main() {  
    let v = vec![1, 2, 3];  
    for x in v {  
        v.push(x);  
    }  
}
```

b)

```
enum Drevo {  
    Prazno,  
    Sestavljeno(Drevo, u32, Drevo),  
}
```

c)

```
fn daljsi_niz(s1: &str, s2: &str) -> &str {  
    if s1.len() > s2.len() {  
        return s1;  
    } else {  
        return s2;  
    }  
}
```

d)

```
fn g(s1: &String, s2: &String) -> () {  
    /// Poljubna koda, da je tip funkcije ustrezen  
}  
  
fn main() {  
    let mut s1 = String::from("1");  
    g(&mut s1, &s1);  
}
```

e)

```
fn vsebuje<T>(v: &Vec<T>, x : &T) -> bool {  
    for y in v {  
        if x == y {  
            return true  
        }  
    }  
    return false  
}
```