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)

```
fn f(a: i32, b: i32) -> i32 {
  a * b
}
fn g(x: i32) -> i32 {
  f(x, x + 1)
}
fn main() {
  let m = 6;
  let n = g(m);
  println!("{n}")
}
```

b)

```
fn f(s: String) {
  println!("{s}")
}
fn g(s: String) {
  f(s)
}
fn main() {
  let s2 = String::from("2");
  let s1 = String::from("4");
  if true {
    println!("{s2}");
  }
  g(s1);
}
```

c)

```
fn f(s: &String) {
  println!("{s}")
}
fn g(s: String) {
  f(&s)
}
fn main() {
  let s1 = String::from("4");
  let s2 = String::from("2");
  g(s1);
  println!("{s2}");
}
```

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 Rust program zavrne;
2. pokažite primer nedefiniranega vedenja, ki ga Rust s tem prepreči (če ga);
3. program popravite tako, da bo veljaven in bo učinkovito dosegel prvotni namen.

a)

b)

c)

d)

e)