

## 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 morebitnega kasnejšega nedefiniranega vedenja, ki ga Rust s tem prepreči, oziroma navedite, da gre za omejenost jezika in da lahko program vedno varno uporabimo;
3. 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 T {  
    A,  
    B(T, u32, T),  
}
```

c)

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