

V oni stari nalogi imamo dva vrča; v vrč **a** gre 7 litrov in v vrč **b** gredo 4 litri. Vodo lahko točimo:

- iz jezera v vrč, pri čemer vrč vedno napolnimo do vrha;
- iz vrča v jezero, pri čemer vrč vedno popolnoma izpraznimo;
- iz vrča v vrč, pri čemer točimo toliko časa, da je vrč, v katerega točimo, poln, ali pa vrč, iz katerega točimo, prazen - karkoli od tega se zgodi prej.

Naloga običajno zahteva, da z vrčema namerimo določeno količino vode, na primer dva litra. Naša ni takšna.

Obvezna naloga

Tole je skoraj bolj naloga iz pogojev kot iz zank.

Zaporedje pretakanj naj bo podano takole (te vrstice skopiraj na začetek programa):

```
poteze = [("j", "a"), ("a", "b"), ("b", "j"), ("a", "b"), ("j", "a"),
          ("a", "b"), ("b", "j"), ("a", "b"), ("a", "j"), ("b", "a"),
          ("j", "a"), ("a", "j")]
```

Pri tem prvi element para pove, odkod točimo, drugi pa, kam. Napiši program, ki simulira takšno pretakanje in sproti poroča, koliko vode je v katerem vrču. Za gornji primer mora izpisati:

```
a: 7 b: 0
a: 3 b: 4
a: 3 b: 0
a: 0 b: 3
a: 7 b: 3
a: 6 b: 4
a: 6 b: 0
a: 2 b: 4
a: 0 b: 4
a: 4 b: 0
a: 7 b: 0
a: 0 b: 0
```

Rešitev

Zelo malo imamo komentirati.

- Shranjevati moramo stanje obeh vrčev; spremenljivki bomo poimenovali **a** in **b**.
- Zanko, s katero gremo čez vrče, najlepše zapišemo kot **for odkod, kam in poteze**; odkod in kam lahko seveda poimenujemo drugače, bistveno je, da gremo čez seznam parov in ga razpakiramo v dve spremenljivki.
- Znotraj zanke moramo obravnavati vse možne poteze. To bomo počeli bodisi s pogojnimi stavki oblike **if odkod == "a" and kam == "b"**, ali

pa ločenima gnezdenima pogojemama, `if odkod == "a"` in znotraj tega `if kam == "b"`. Spodnji program uporablja slednje.

- Pri nekaterih potezah moramo preverjati, ali prej zmanjka vode ali posode. Za to poskrbi še en `if`.
- Pri nekaterih `else` sem v komentar, ki mu sledi, dopisal, kateri primer pokriva. To je dobra praksa, ki poveča čitljivost programa.

```
poteze = [("r", "a"), ("a", "b"), ("b", "j"), ("a", "b"), ("j", "a"), ("a", "b"),  
          ("b", "j"), ("a", "b"), ("a", "j"), ("b", "a"), ("j", "a"), ("a", "j")]
```

```
a = b = 0  
for odkod, kam in poteze:  
    if odkod == "a":  
        if kam == "b":  
            if b + a > 4:  
                a -= 4 - b  
                b = 4  
            else:  
                b += a  
                a = 0  
        else:  
            a = 0  
    elif odkod == "b":  
        if kam == "a":  
            if b + a > 7:  
                b -= 7 - a  
                a = 7  
            else:  
                a += b  
                b = 0  
        else:  
            b = 0  
    else: # odkod == "j"  
        if kam == "a":  
            a = 7  
        else: # kam == "b"  
            b = 4  
    print("a:", a, "b:", b)
```

```
a: 7 b: 0  
a: 3 b: 4  
a: 3 b: 0  
a: 0 b: 3  
a: 7 b: 3  
a: 6 b: 4  
a: 6 b: 0  
a: 2 b: 4
```

```
a: 0 b: 4
a: 4 b: 0
a: 7 b: 0
a: 0 b: 0
```

Se da kaj poenostaviti? Pogojev pri pretakanju med posodama se da izogniti z uporabo funkcije `min`: pretočimo toliko, kolikor je v `a` ali toliko, kolikor gre v `b`, kar je pač manjše.

```
poteze = [("r", "a"), ("a", "b"), ("b", "j"), ("a", "b"), ("j", "a"), ("a", "b"),
          ("b", "j"), ("a", "b"), ("a", "j"), ("b", "a"), ("j", "a"), ("a", "j")]
```

```
a = b = 0
for odkod, kam in poteze:
    if odkod == "a":
        if kam == "b":
            kolicina = min(a, 4 - b)
            a, b = a - kolicina, b + kolicina
        else:
            a = 0
    elif odkod == "b":
        if kam == "a":
            kolicina = min(b, 7 - a)
            a, b = a + kolicina, b - kolicina
        else:
            b = 0
    else: # odkod == "j"
        if kam == "a":
            a = 7
        else: # kam == "b"
            b = 4
    print("a:", a, "b:", b)
```

```
a: 7 b: 0
a: 3 b: 4
a: 3 b: 0
a: 0 b: 3
a: 7 b: 3
a: 6 b: 4
a: 6 b: 0
a: 2 b: 4
a: 0 b: 4
a: 4 b: 0
a: 7 b: 0
a: 0 b: 0
```

Morda pa je preprosteje, če obravnavamo posamične primere brez gnezdenja pogojev? In celo če, v nasprotju z vsem, kar poskušam študentom vbiti v glavo,

ne razpakiramo terke?

```
a = b = 0
for poteza in poteze:
    if poteza == ("j", "a"):
        a = 7
    if poteza == ("j", "b"):
        b = 7
    if poteza == ("a", "j"):
        a = 0
    if poteza == ("b", "j"):
        b = 0
    if poteza == ("a", "b"):
        a, b = a - min(a, 4 - b), b + min(a, 4 - b)
    if poteza == ("b", "a"):
        a, b = a + min(b, 7 - a), b - min(b, 7 - a)
    print("a:", a, "b:", b)
```

```
a: 0 b: 0
a: 0 b: 0
a: 0 b: 0
a: 0 b: 0
a: 7 b: 0
a: 3 b: 4
a: 3 b: 0
a: 0 b: 3
a: 0 b: 3
a: 3 b: 0
a: 7 b: 0
a: 0 b: 0
```

Hm, pravzaprav je bilo tole res preprosteje.

V Pythonu 3.10 pa lahko uporabimo novi stavek `match`, ki program še malo skrajša.

```
a = b = 0
for poteza in poteze:
    match poteza:
        case "j", "a":
            a = 7
        case "j", "b":
            b = 7
        case "a", "j":
            a = 0
        case "b", "j":
            b = 0
        case "a", "b":
            a, b = a - min(a, 4 - b), b + min(a, 4 - b)
        case "b", "a":
            a, b = a + min(b, 7 - a), b - min(b, 7 - a)
```

```

        a, b = a - min(a, 4 - b), b + min(a, 4 - b)
    case "b", "a":
        a, b = a + min(b, 7 - a), b - min(b, 7 - a)
    print("a:", a, "b:", b)

a: 0 b: 0
a: 0 b: 0
a: 0 b: 0
a: 0 b: 0
a: 7 b: 0
a: 3 b: 4
a: 3 b: 0
a: 0 b: 3
a: 0 b: 3
a: 3 b: 0
a: 7 b: 0
a: 0 b: 0

```

Dodatna naloga

Zdaj pa obratno: imamo zaporedje stanj vrčev, podanih s seznamom terk, na primer.

```
stanja = [(7, 0), (3, 4), (3, 0), (0, 3), (7, 3), (6, 4),
          (6, 0), (2, 4), (0, 4), (4, 0), (7, 0), (0, 0)]
```

Napiši program, ki za takšno zaporedje izpiše pripadajoči vrstni red pretakanj. Izpis naj bo v takšni obliki:

```

j a
a b
b j
a b
j a
a b
b j
a b
a j
b a
j a
a j

```

Rešitev

Osnovna ideja je, da moramo primerjati prejšnje in novo stanje ter izpisati ustrezno spremembo.

```
seq = [(7, 0), (3, 4), (3, 0), (0, 3), (7, 3), (6, 4), (6, 0), (2, 4), (0, 4), (4, 0), (7, 0), (0, 0)]
a = b = 0

```

```

for na, nb in seq: # na in nb sta novi a in novi b
    if a == na: # a je ostal enak, nekaj se dogaja z b
        if nb == 0:
            print("b j")
        else:
            print("j b")
    elif b == nb: # b je ostal enak, nekaj se dogaja z a
        if na == 0:
            print("a j")
        else:
            print("j a")
    else: # sicer točimo med posodama
        if na > a: # količina v a se je povečala ...
            print("b a")
        else:
            print("a b")
a, b = na, nb

j a
a b
b j
a b
j a
a b
b j
a b
a j
b a
j a
a j

```