Kaprekarjevo zaporedje

Vhodne podatke preberete v zanki s standardnega vhoda in rezultate izpišite na standardni izhod.

```
if __name__ == '__main__':
    import sys
    for line in sys.stdin:
        print(kaprekar(int(line))[-1])
```

Postopek, ki za dani člen Kaprekarjevega zaporedja poišče naslednji člen:

1. Najprej določimo nov člen zaporedja, tako da števke v zapisu danega člena uredimo v padajočem vrstnem redu.
2. Nato poiščemo število, ki ima števke v zapisu danega člena urejene v naraščajočem zaporedju.
3. Naslednji člen zaporedja je razlika med njima.

Vhod

S standardnega vhoda preberite število n. Tvorite člene zaporedja $k_n = f(k_{n-1})$, dokler se prvič ne ponovi število k' v zaporedju.

Vzamete število n za začetno vrednost in tvorite novo zaporedje, dokler se v zaključku zanke spet ne ponovi število k'.

Izhod

Izpišite člene v zanki, ločene s presledkom, na standardni izhod.

Primer

Vhod	Izhod	Pojasnilo
`123430`	`420876 851742 750843 840852 860832 862632 642654`	$k = 420876$
`23415`	`82962 75933 63954 61974`	$k = 23415$

Koda TAKO

Vhodne podatke preberete v zanki s standardnega vhoda in izpišete na standardni izhod `True` oziroma `False`, odvisno od tega, ali je prišlo do napake ali ne.

```
if __name__ == '__main__':
    import sys
    for nim in sys.stdin:
        print(check_isbn(nim.strip()))
```

Kodo `TAKO` sestavlja 10 števk. Običajno se zapiše v obliki

`x-xxx-xxxxx-x`, kjer smo z `x` označili števke 0–9.

Štiri skupine števk so ločene z znakom `'-'`. Prva skupina vsebuje eno samo števko, sledi skupina treh in petih števk, na koncu je dodana še ena števka. Prvih 9 števk je poljubnih, zadnja pa je z njimi določena in jo imenujemo *preizkusno vsoto*.

Če se pri prepisovanju kode zgodi napaka, lahko v večini primerov to ugotovimo.

Zadnja števka, preizkusna vsota, je izbrana tako, da je vsota produktov števil, ki pripadajo števkam po vrsti od leve proti desni s števili [10, 9, 8, 7, 6, 5, 4, 3, 2, 1], deljiva z 11. Za kodo `0-306-40615-2` izračunamo vsoto

```
10*0 + 9*3 + 8*0 + 7*6 + 6*4 + 5*0 + 4*6 + 3*1 + 2*5 + 2 = 132
```

Število 132 je deljivo z 11. `132 = 11 * 12`. Sklepamo, da do napake ni prišlo.

Definicija problema

Napiši funkcijo, ki preveri pravilnost vpisane kode `TAKO`.

Primera

1. primer

Vhod

```
0-306-40615-2
```

Izhod

```
True
```

2. primer

Vhod

```
0-306-40615-3
```

Izhod

```
False
```

Filter praštevil

Vhodne podatke preberete v zanki s standardnega vhoda in rezultate izpišite na standardni izhod.

```
if __name__ == '__main__':
    import sys
    for line in sys.stdin:
        print(sum(prime_filter(eval(line))))
```

Iščemo praštevila v seznamu naključno izbranih števil.

Definicija problema

Iz seznama naravnih števil izločite praštevila.

Vhodni podatki

V zanki preberite s standardnega vhoda nize z zapisom seznamov števil.
Niz pretvorite v seznam s funkcijo `eval`.
Na izhod izpišite vsoto praštevil v seznamu.

Primer

Vhod

```
[94, 32, 49, 97, 90, 61, 69, 9, 53]
```

Izhod

```
211
```

Izštevanka 7

Vhodne podatke preberete v zanki s standardnega vhoda in izpišite rezultate na standardni izhod.

```
if __name__ == '__main__':
    import sys
    for line in sys.stdin:
        print(izstevanka7(int(line)))
```

V znani družabni igri poštevanka 7 smo sedeli v krogu in šteli.

Določili smo prvega igralca in smer, v kateri bo potekala igra. Prvi igralec je zaklical ena, naslednji je zaklical dve in tako naprej v krogu. Vendar pa to ni bilo običajno štetje. Če si bil na vrsti, da zakličeš število, ki je deljivo s 7 ali če se v desetiškem zapisu tega števila nahaja števka 7, nisi smel izgovoriti števila, ampak si moral zaklicati *bum*. Če si se zmotil, si storil napako, in kazen je bila izločitev iz igre.

Definicija problema

V našem primeru bomo uporabili poštevanko kot odštevanko. Razporedimo n otrok v krog. Zaradi enostavnosti označimo otroke s številkami od 0 do $n-1$. Začnemo s štetjem pri otroku, ki smo ga označili s številko 0. Štejemo do prvega otroka, pri katerem pride na vrsto *bum*. Otroka izločimo in nadaljujemo štetje. Če štetje preseže 99, se nadaljuje spet pri 1. Igro ponavljamo toliko časa, da ostane en sam otrok.

Vhodni podatki

Program sprejema število otrok n.

Izhodni podatki

Številka, s katero je označen otrok, ki ostane na koncu.

Primeri

Vhod	Izhod
67	17
19	4
61	3

Verižni ulomki

Vhodne sezname preberete v zanki s standardnega vhoda in izpišete števec in imenovalec na standardni izhod.

```
if __name__ == '__main__':
    import sys
    for line in sys.stdin:
        print(verizni(eval(line)))
```

Verižni ulomek

Ulomek zapišemo v kot seznam koeficientov verižnega ulomka.

$$\frac{a}{b} = [a_0, a_1, a_2, a_3, \dots] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots}}}}$$

Primer razvoja v verižni ulomek

- $\frac{415}{93} = 4 + \frac{43}{93} = 4 + \frac{1}{\frac{93}{43}}$,
- $\frac{93}{43} = 2 + \frac{7}{43}, \rightarrow \frac{415}{93} = 4 + \frac{1}{2 + \frac{1}{\frac{43}{7}}}$,
- $\frac{415}{93} = 4 + \cfrac{1}{2 + \cfrac{1}{6 + \cfrac{1}{7}}} = [4; 2, 6, 7]$.

Primeri

Vhod	Izhod
`[0, 1, 2, 1, 5]`	`17 23`
`[1, 1, 1, 1, 1, 1, 1, 1]`	`34 21`
`[0, 1, 1, 1, 1, 1, 1, 1]`	`13 21`