1	2	3	4	5	6	7	8	9	10

Neprajem si zverejniť moje výsledky testu na webe: _____

Test z Programovania pre 1. ročník

1. Druhý parameter funkcie **`xy()`** je nejaký zoznam celých čísel:

```
def xy(ret, ix, znaky):
    p = list(ret)
    for i in range(len(ix)):
        p[ix[i]] = znaky[0]
        znaky = znaky[1:]
    return ''.join(p)

>>> xy('najpython', (_____), 'anopye')
'neopytany'
```

   Zistite, aký bol druhý parameter pri volaní tejto funkcie.

2. Príkaz **`input()`** sme testovali v príkazovom režime::

```
>>> x = input(input(input('zadaj:')))
zadaj:preco:
preco:lebo:
lebo:nic:
>>>
```

   Zistite, čo sa takto priradilo do premennej **`x`**.

3. Funkcia **`generuj()`** vytvára nejaký znakový reťazec:

```
def generuj(n, retazec):
    for i in range(n):
        retazec = 'a\nb' + retazec[1:-1] + 'x\ny' + retazec[2:-2] + 'b\na'
    return retazec

print(generuj(2, 'xyz'))
```

   Po zavolaní tejto funkcie sa vypíše niekoľko riadkov textu. Zistite, koľko riadkov sa vypíše a koľkokrát sa v nich vyskytnú písmená **`'a'`** a **`'b'`**.

4. Funkciu **`kresli()`** sme zavolali s parametrom pole dvojíc:

```
def kresli(bodky):
    for x,y in bodky:
        if x < y:
            farba = 'red'
        elif x+y < 100:
            farba = 'blue'
        else:
            farba = 'yellow'
        canvas.create_oval(x-2, y-2, x+2, y+2, fill=farba, outline='')

kresli([(80, 70), (30, 90), (80, 50), (70, 40), (70, 50), (50, 10), (10, 10),
        (40, 90), (40, 10), (80, 20), (60, 20), (50, 60), (40, 50), (20, 40),
        (20, 70), (90, 80), (70, 90), (50, 20), (30, 70)])
```

   Funkcia kreslí farebné bodky. Zistite, koľko bodiek bolo červených, koľko modrých a koľko žltých?

5. Funkcia **`zapis()`** vytvára textový súbor:

```
def zapis(subor, n):
    with open(subor, 'w') as vystup:
        j = 3
        for i in range(n):
            if i < j:
                ret = ' '
            else:
                ret = '\n'
                j += i
            print(i, i*i, end=ret, file=vystup)
        print(file=vystup)
```

Opravte túto funkciu tak, aby pracovala presne rovnako, ale nevolala štandardnú funkciu **`print()`**.

6. Funkcia **`ntica()`** vytvára nejakú n-ticu dvojíc celých čísel:

```
def ntica(a, b):
    vysl = ((a,b),)
    while b > 0:
        a, b = b, a%b
        vysl += ((a,b),)
    return vysl
```

Takto vytváraná postupnosť dvojíc čísel má nejaký súvis s výpočtom najväčšieho spoločného deliteľa dvoch celých čísel. Dopíšte:.

```
def nsd(a, b):
    x = ntica(a, b)

    return _________________________
```

7. Funkcia **`urob()`** s tromi parametrami spracuje nejaké tri postupnosti (napr. sú to tri polia, n-tice alebo reťazce):

```
def urob(a, b, c):
    return c[-1:len(c)] + b[1:] + a[:] + c[:-1] + b[0:1]
```

Do premennej p sme priradili nejakú hodnotu a zavolali s ňou funkciu **`urob()`** takto:.

```
>>> p = ___________________________
>>> urob(p[2:7], p[4:], p[:5])
(4, 9, 6, 11, 10, 5, 8, 3, 4, 9, 6, 7, 12, 8, 3, 4)
```

Zistite, s akou hodnotou premennej **`p`** sme zavolali túto funkciu.

8. Dopíšte funkciu **`porovnaj()`**, ktorá porovná dva dátumy (dátum je trojica celých čísel (deň, mesiac, rok)) a vráti jeden z týchto troch reťazcov: **`'mensi'`**, **`'rovny'`**, **`'vacsi'`**:

```
def porovnaj(datum1, datum2):
```

```
>>> porovnaj((26, 10, 2015), (15, 11, 2015))
'mensi'
>>> porovnaj((26, 10, 2015), (15, 11, 2014))
'vacsi'
>>> porovnaj((26, 10, 2015), (26, 10, 2015))
'rovny'
```

9. Funkcia **`rozsekaj()`** vytvorí zo znakového reťazca nticu podreťazcov, pričom oddeľovačom týchto podreťazcov v pôvodnom reťazci je znak bodkočiarka `';'`. Funkcia je podobná štandardnej metóde **`split()`**, ktorú ale požiť nesmiete:

```
def rozsekaj(retazec):
    vysl = ()
    while retazec:
        i = ';'.index(retazec+';')
        vysl.append(retazec[:i])
        retazec[i+1:] = ''
    return vysl
```

```
>>> rozsekaj('a;12;b')
('a', '12', 'b')
```

Vo funkcii sú chyby, opravte ich. Nedopisujte nové príkazy.

10. Funkcia **`uprac()`** nejako mení obsah poľa:

```
def uprac(pole):
    i = 0
    for j in range(len(pole)):
        if pole[j] < 0:
            pole.insert(i, pole.pop(j))
            i += 1
```

Prerobte túto funkciu tak, aby pracovala nielen pre polia, ale aj pre n-tice. Preto nebude táto funkcia modifikovať svoj parameter, ale bude vytvárať novú nticu a tú vráti ako výsledok funkcie:

```
def uprac(pole):
    i = 0
    pole = tuple(pole)
    for j in range(len(pole)):
        if pole[j] < 0:
```

```
            i += 1
    return pole
```

Doplňte chýbajúcu časť funkcie (zvyšok funkcie už nemeňte!).