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. Zistite, čo vypočíta daná rekurzívna funkcia:

```
def pocitaj(n):
    if n <= 1:
        return list(range(n+1))
    pred = pocitaj(n-1)
    return pred + [sum(pred[-2:])]

print(pocitaj(10))
```

2. Máme zadefinovať funkciu **`ocisluj(pole)`**, ktorá zmení všetky prvky dvojrozmerného poľa tak, že ich postupne prechádza po stĺpcoch a čísluje ich celými číslami od 0, pritom riadky poľa nemusia mať rovnakú dĺžku, napr.

```
>>> ab = [[1, 1], [1, 1, 1], [1], [1, 1, 1, 1]]
>>> ocisluj(ab)
>>> ab
[[0, 4], [1, 5, 7], [2], [3, 6, 8, 9]]
```

Dopíšte chýbajúce časti funkcie

```
def ocisluj(pole):
    m = 0
    for r in pole:
        m = ______________________________
    poc = 0
    for j in range(m):
        for i in range(len(pole)):
            if _________________________________:
                pole[i][j] = poc
                poc += 1
```

3. Funkcia **`zisti()`** pracuje s asociatívnym poľom:

```
def zisti(pole):
    asoc, neviem = {}, None
    for prvok in pole:
        asoc[prvok] = asoc.get(prvok, 0) + 1
        if neviem is None or asoc[prvok] > asoc[neviem]:
            neviem = prvok
    return neviem
```

Zistite, čo sa vypíše pre volania:

```
print(zisti([1,2,3,4,5,4,3,2,3,2,4,6,4]))
print(zisti(list('programujem v pythone')))
```

4. V triede **`Mnozina`** je 5 chýb. Opravte ich! Neopravujte kód, ktorý nie je chybný.

```
class Mnozina:
    def __init__(self):
        self.pole = []

    def __repr__(self):
        print(tuple(sorted(self.pole)))

    def add(self, cislo):
        if cislo in self:
            self.pole.append(cislo)

    def discard(self, cislo):
        if cislo in self:
            self.pole.pop(cislo)

    def __contains__(self, cislo):
        return self.pole.index(cislo) >= 0

    def __len__(self):
        return len(self.pole)

    def zjednotenie(self, ina):
        for i in ina.pole:
            self.append(i)
```

5. Dopíšte funkciu **`nechaj_float(pole)`**, ktorá v poli znakových reťazcov ponechá len tie, ktoré reprezentujú desatinné čísla (dajú sa previesť konverznou funkciou **`float()`**). Dopisuje len medzi riadky **`while`** a **`del`**:

```
def nechaj_float(pole):
    i = 0
    while i < len(pole):




            del pole[i]

>>> pole = ['3..', '1e1', '7', '', '1e1e', '.7.']
>>> nechaj_float(pole)
>>> pole
['1e1', '7']
```

6. **a)** Na začiatku bol zásobník prázdny. Potom sme vykonali niekoľko operácií **`push`** a **`pop`**. Zistite, čo sa vypíše po spustení:

```
print('pre zasobnik:')
for x in 7,3,5,11,3,8,4:
    push(x)
for x in range(6):
    print(pop(), pop(), end=' ')
    push(x)
print(pop())
```

b) Na začiatku bol rad prázdny. Potom sme vykonali niekoľko operácií **`enqueue`** a **`dequeue`**. Zistite, čo sa vypíše po spustení:

```
print('pre rad:')
for x in 7,3,5,11,3,8,4:
    enqueue(x)
for x in range(6):
    print(dequeue(), dequeue(), end=' ')
    enqueue(x)
print(dequeue())
```

7. Nasledovný príklad definície triedy demonštruje nie najlepšie využitie operátora indexovania. Zistite ale, čo sa vypíše:

```
class Trieda:
    x = 7
    def __getitem__(self, i):
        return sum(range(self.x-i))
    def __setitem__(self, i, y):
        self.x = 2*i+y

t = Trieda()
print(t[4], t[5])
t[4] = 1
print(t[4], t[5])
```

8. Preveďte **infixový** výraz do **prefixu** a aj do **postfixu**:

```
3*(4+5)-6*7/(8+6)+(2/3)/(4/5)
```

prefix:

postfix:

9. Funkcia **`vsetky()`** nejako spracováva pole množín:

```
def vsetky(pole_mnozin):
    vysl, b = set(), True
    for m in pole_mnozin:
        if b:
            vysl |= m
        else:
            vysl -= m
        b = not b
    return vysl
```

Zistite, čo treba dosadiť do premennej **`x`**, aby sem dostali tento výsledok:

```
>>> x = _________________________
>>> vsetky([x, set(range(3,10,2)), set(range(5,15,3))])
{2, 5, 6, 8, 11, 14}
```

10. Napíšte program pre Turingov stroj, ktorý bude akceptovať vstup na páske len vtedy, keď je na nej číslo v dvojkovej sústave, pričom toto číslo je nepárne. Štartový stav má meno `'0'` a koncový `'1'`:

```
t = t.Turing('''









''')
for slovo in '', '11a', '0101010101', '11110', '1', '0':
    t.znovu(slovo)
    print(repr(slovo), t.rob(False))
```

Vypíše:

```
'' False
'11a' False
'0101010101' True
'11110' False
'1' True
'0' False
```