1	2	3	4	5	$\sum$

Informatika 2, 3. ZH (2019-05-13)

A feladatok megoldása elfér a feladat mellett, ha külön lapra írjuk, tegyünk egy jól látható nyilat a helyére!

1.

a) Írjuk meg a faktoriális függvényt, de ne rekurzívan! *(2 pont)*

```
def factorial(n):
```

b) Írjunk egy olyan függvényt, ami egy listát és egy számot kap és a bemenetül kapott lista minden elemét megszorozza azzal a számmal. Ne adjon vissza semmit a függvény, de változtassa meg az eredeti listát. *(2 pont)*

```
def szoroz(l, n):
```

c) Írjunk egy olyan függvényt ami egy pozitív és két nemnegatív számot kap (n, i és j) és visszaad egy olyan $n \times n$-es `numpy` mátrixot, aminek az i-edik sorának j-edik eleme 1, a többi 0. *(2 pont)*

```
def E(n, i, j):
```

2. Elméleti kérdések *(4 pont)*

a) Hányféle képpen bontható fel a 3 pozitív egészek összegére, ha a sorrend is számít?

b) Legfeljebb hány lépésben keres meg egy elemet egy n-hosszú rendezett listában a bináris keresés?

c) Mondjunk példát egy nem referencia típusra!

d) Mi az a prefix jelölés (lengyel forma)?

3. Jobb oldalt látható az alap számológép kódja, de pár hiba van benne, algoritmikus és szintaktikai is. Találjuk meg ezeket és javítsuk ki! *(4 pont)*

```
class Node(object):
    def __init__(self, kappa):
        i = -1
        if kappa.find("+") != -1:
            i = kappa.find("+")
        elif kappa.find("-") != -1:
            i = kappa.find("-")
        elif kappa.find("*") != -1:
            i = kappa.find("*")
        elif kappa.find("/") != -1:
            i = kappa.find("/")

        if i != -1
            self.data = kappa[i]
            self.left = Node(kappa[:i])
            self.right = Node(kappa[i + 1:])
        else:
            self.data = kappa
            self.left = None
            self.right = None

    def calculate(self):
        if self.data == "+":
            return self.left.calculate() - \
                   self.right.calculate()
        elif self.data == "-":
            return self.left.calculate() + \
                   self.right.calculate()
        elif self.data == "*":
            return self.left.calculate() * \
                   self.right.calculate()
        elif self.data == "/":
            return self.left.calculate() / \
                   self.right.calculate()
        else:
            return self.data
```

4. Írjunk egy olyan függvényt, ami egy sztringet kap és igaz/hamis-ad ad vissza attól függően, hogy a sztring jól van-e zárójelezve. *(3 pont)*

Akkor van egy kifejezés jól zárójelezve, ha bármely pontján igaz, hogy legalább annyi zárójel van kinyitva, mint amennyi addig bezárva, valamint a végére pont ugyanannyi legyen bezárva, mint kinyitva.

```
def jozarojel(s):
```

Példa:

```
>>> jozarojel("1/(3*(3-1))")
True
>>> jozarojel("1/(3*2))-1(")
False
```

5. Írjuk meg a Hanoi tornyai játék megoldását rekurzív algoritmussal. *(3 pont)*

```
def hanoi(n, honnan, hova, minat):
```

Példa:

```
>>> hanoi(3,"A","B","C")
A ==> B
A ==> C
B ==> C
A ==> B
C ==> A
C ==> B
A ==> B
```