Ocenjevanje: Python

Ocenjuje se tudi napisana koda, če tudi ne deluje pravilno.

1/3 naloge, če je dojel kaj je potrebno narediti,

2/3 naloge, če je pravilno predstavil algoritem

3/3 naloge, če je naredil delujoč program

Skupaj 100% za vse naloge, ki jih mora rešiti.

1-skupina I 7 razred ena naloga velja 25 %

2-skupina J 8 razred ena naloga velja 20 %

3-skupina K 9 razred ena naloga velja 17 % zaokroženo

Informacije Borut Jurčič Zlobec

Email: borut.jurcic.zlobec@gmail.com

tel. 041 41 82 89

REŠITVE:

```
#!/usr/bin/env python3
from math import sqrt
import locale
locale.setlocale(locale.LC_ALL, 'sl_SI.UTF-8')
import json

#1. Uravnotežen trojiški sistem
def tri2dec(y):
    tri_digit = {'+': 1, 'o': 0, '-': -1}
    res = 0
    b = 3
```

```
    for ch in y:
        res = res * b + tri_digit[ch]
    return res

#2. Uravnotežen trijiški sistem drugič
def dec2tri(x):
    digit_tri = {1: '+', 0: 'o', -1: '-'}
    res = ''
    while x:
        a = x % 3
        if a == 2:
            a = -1
            x += 1
        res += digit_tri[a]
        x = x // 3
    return res[::-1]

#3. Faktorizacija
def factors(n):    # (cf. https://stackoverflow.com/a/15703327/849891)
    j = 2
    while n > 1:
        for i in range(j, int(sqrt(n + 0.05)) + 1):
            if n % i == 0:
                n //= i ; j = i
                yield i; break
        else:
            if n > 1:
                yield n; break

#4. Podnizi
def podnizi_all(a_str, sub):
```

```
    start = 0
    while True:
        start = a_str.find(sub, start)
        if start == -1: return
        yield start
        start += len(sub) # use start += 1 to find overlapping matche

#5. Frekvence znakov v nizu
def freq_ch(a_str):
    f = {}
    for ch in a_str:
        f[ch] = f.get(ch, 0) + 1
    return f

#6. Obrazila
def suffix(s):
    res = []
    for i in range(len(s)):
        res.append(s[i:])
    res = sorted(list(enumerate(res)), key = lambda x: x[1])
    return res

#7. Trie
class Trie(object):
    def __init__(self):
        self.child = {}

    def insert(self, word):
        current = self.child
        for l in word:
            current[l] = current.get(l, {})
```

```
            current = current[l]
        current['$'] = {}
    def __str__(self): return json.dumps(self.child)

if __name__ == '__main__':
    n = int(input("n = "))
    print(list(factors(n)))

    ob = Trie()
    ws = ['jabolko', 'jablana', 'hruska']
    for w in  ws:
        ob.insert(w)
    print(ob)

    ob = Trie()
    ws = ['be', 'bee', 'can', 'cat', 'cd']
    for w in  ws:
        ob.insert(w)
    print(ob)
```