Zadatci za vježbu

RJEŠENJA

Zadatak 1.

File Edit Format Run Options Windows Help

```
# 02_01
from random import *
from time import *

def najmanji(a):
    najm = a[0]
    for t in a:
        if t < najm:
            najm = t
    return najm

v = [1000, 10000, 100000]

for t in v:
    a = [randint(1, t * 10) for i in range(t)]
    poc = clock()
    k = najmanji(a)
    kraj = clock()
    print('{0} elemenata kreirana funkcija {1:7.5f} ms'.format(t, (kraj - poc) * 1000))
    poc = clock()
    k = min(a)
    kraj = clock()
    print('{0} elemenata ugrađena funkcija {1:7.5f} ms'.format(t, (kraj - poc) * 1000))
```

Zadatak 2.

a.

```
File Edit Format Run Options Windows Help
# 02_02_a
def f(n):
    if n == 1:
        return 1
    else:
        return 1 / n + f(n - 1)

n = int(input())
print('{0:4.2f}'.format(f(n)))
```

b.

```
File Edit Format Run Options Windows Help
# 02_02_b
def f(n):
    if n == 1:
        return 1
    else:
        return f(n - 1) +  1 / (n * n)

n = int(input())
print('{0:4.2f}'.format(f(n)))
```

c.

```
File Edit Format Run Options Windows Help
# 02_02_c
def f(n):
    if n == 1:
        return 1
    elif n % 2 == 0:
        return f(n - 1) - n
    else:
        return f(n - 1) + n

n = int(input())
print('{0}'.format(f(n)))
```

d.

```
File Edit Format Run Options Windows Help
# 02_02_d
def f(n):
    if n == 1:
        return 1
    elif n % 2 == 0:
        return f(n - 1) - n / (n + 1)
    else:
        return f(n - 1) + n / (n + 1)

n = int(input())
print('{0:4.2f}'.format(f(n)))
```

e.

```
File Edit Format Run Options Windows Help
# 02_02_e
def f(n):
    if n == 1:
        return 1
    elif n % 2 == 0:
        return f(n - 1) - 1 / (2 * n - 1)
    else:
        return f(n - 1) + 1 / (2 * n - 1)

n = int(input())
print('{0:4.2f}'.format(f(n)))
```

Zadatak 3.

```
File Edit Format Run Options Windows Help
# 02_03
def f(a, n):
    if n == 1:
        return a
    else:
        return a * f(a, n - 1)

a = int(input())
n = int(input())
print(f(a, n))
```

Zadatak 4.

```
File Edit Format Run Options Windows Help
# 02_04
def f(n):
    if n == 1:
        return '1'
    else:
        return f(n // 2) + str(n % 2)

n = int(input())
print(f(n))
```

Zadatak 5.

```
File Edit Format Run Options Windows Help
# 02_05
def f(s, c):
    if len(s) == 0:
        return 0
    elif s[0] == c:
        return f(s[1:], c) + 1
    else:
        return f(s[1:], c)

s = input()
c = input()
print(f(s, c))
```

Zadatak 6.

```
File Edit Format Run Options Windows Help
# 02_06
def f(n):
    if n == 1:
        return 6 ** 0.5
    else:
        return (f(n - 1) + 6) ** 0.5

n = int(input())
print('{0:5.2f}'.format(f(n)))
```

Zadatak 7.

```
File Edit Format Run Options Windows Help
# 02_07
def f(a):
    if len(a) == 1:
        return a[0]
    else:
        t = f(a[1:])
        if t > a[0]:
            return t
        else:
            return a[0]

n = int(input())
a = []
for i in range(n):
    a.append(int(input()))
print(f(a))
```

Zadatak 8.

File Edit Format Run Options Windows Help

```
# 02_08
# slijedi kraće objašnjenje
#
##U n-terokutu fiksiramo jednu stranicu i gledamo sve trokute koje možemo
##kreirati nad tom strnicom.
##Ilustrirajmo to primjerice na 8-rokutu. Neka su vrhovi 8-rokuta
##označeni redom brojevima 1, 2, 3, 4, 5, 6, 7, 8
##Fiksirajmo primjerice stranicu određenu vrhovima 1 i 2. Tada
##treći vrh trokuta može biti neki od sljedećih vrhova: 3, 4, 5, 6, 7 ili 8.
##Promotrimo što dobivamo odabirom pojedinog od preostalih vrhova:
##4 - jedan trokut (2, 3, 4) i jedan šesterokut: (1, 4, 5, 6, 7, 8)
##5 - jedan četverokut (2, 3, 4, 5) i jedan peterokut (1, 5, 6, 7, 8)
##6 - jedan peterokut(2, 3, 4, 5, 6) i jedan četverokut (1, 6, 7, 8)
##Kao što vidimo u principu dobivamo likove kod kojih je ukupni zbroj
##vrhova u našem slučaju 9, a inače općenito n + 1.
##Ako imamo n-terokut i jedan od likova na koji trokut dijeli n-terokut je
##k-terokut, drugi će biti (n - k + 1)-terokut. Iz odabira trećeg vrha trokuta
##možemo primijetiti da ukoliko je primjerice odabrana stranica trokuta
##s vrhovima 1 i 2 onda treći vrh može biti i 3. U tom slučaju n-terokut
##podijeljen na (n - 1)-terokut i dvokut te i tu kombinaciju trebamo uzeti
##u obzir te ćemo uzeti da je broj načina na koje se dvokut može podijeliti
##na trokute jednak 1, isto tako je broj načina na koje se trokut može podijeliti
##na trokute također jedan.
##Iz svega navedenog proizlazi da ako je T(n) broj načina na koji se
##n-terokut može podijeliti na trokute, onda vrijedi sljedeća
##jednakost: T(n) = T(2) * T(n - 1) + T(3) * T(n - 2) + ... + T(n - 1) * T(2)
def f(n):
    if n == 3 or n == 2:
        return 1
    else:
        s = 0
        for i in range(2, n):
            s += f(i) * f(n - i + 1)
        return s

n = int(input())
print(f(n))
```

Zadatak 9.

File Edit Format Run Options Windows Help

```
# 02_09
from time import *
from functools import wraps

def pamti(funk):
    izračunano = {}
    @wraps(funk)
    def wrap(*args):
        if args not in izračunano:
            izračunano[args] = funk(*args)
        return izračunano[args]
    return wrap

@pamti
def f1(n):
    if n == 1:
        return 0
    elif n % 2 == 0:
        return f1(n // 2) + 1
    else:
        return f1(3 * n + 1) + 1


def f2(n):
    if n == 1:
        return 0
    elif n % 2 == 0:
        return f2(n // 2) + 1
    else:
        return f2(3 * n + 1) + 1

n = int(input())
poc = clock()
f1(n)
kraj = clock()
print('Trajanje funkcije s pamćenjem: {0}ms'.format((kraj - poc) * 1000))
poc = clock()
f2(n)
kraj = clock()
print('Trajanje obične rekurzije: {0}ms'.format((kraj - poc) * 1000))
```

Zadatak 10.

```
File Edit Format Run Options Windows Help
# 02_10
from time import *
from functools import wraps

def pamti(funk):
    izračunano = {}
    @wraps(funk)
    def wrap(*args):
        if args not in izračunano:
            izračunano[args] = funk(*args)
        return izračunano[args]
    return wrap

@pamti
def f1(n, k):
    if n == 1 or k == 1 or k == n:
        return 1
    else:
        return f1(n - 1, k - 1) + f1(n - 1, k)


def f2(n, k):
    if n == 1 or k == 1 or k == n:
        return 1
    else:
        return f1(n - 1, k - 1) + f1(n - 1, k)

n = int(input())
k = int(input())
poc = clock()
f1(n, k)
kraj = clock()
print('Trajanje funkcije s pamćenjem: {0}ms'.format((kraj - poc) * 1000))
poc = clock()
f2(n, k)
kraj = clock()
print('Trajanje obične rekurzije: {0}ms'.format((kraj - poc) * 1000))
```

Zadatak 11.

File Edit Format Run Options Windows Help

```
# 02_11
from time import *
from random import *
def pretraživanje(a, t):
    for k in a:
        if k == t:
            return True
    return False

def pretraživanje_2(a, t):
    a.sort()
    p = 0
    k = len(a) - 1
    početak, kraj = 0, len(a) - 1
    while početak <= kraj:
        sredina = početak + (kraj - početak) // 2
        if a[sredina] == t:
            return True
        elif a[sredina] > t:
            kraj = sredina - 1
        else:
            početak = sredina + 1
    return False

velicine = [1000, 10000, 100000]
for t in velicine:
    a = [randint(1, t * 10) for i in range(t)]
    k = randint(1, t * 10)
    poc = clock()
    pretraživanje(a, k)
    kraj = clock()
    print(t)
    print('=' * 10)
    print('Slijedno traženje: {0}ms'.format((kraj - poc) * 1000))
    poc = clock()
    pretraživanje_2(a[:], k)
    kraj = clock()
    print('Binarno traženje nakon sortiranja: {0}ms'.format((kraj - poc) * 1000)
    poc = clock()
    k in a
    kraj = clock()
    print('Ugrađena operacija in: {0}ms'.format((kraj - poc) * 1000))
```

Zadatak 12.

```
File Edit Format Run Options Windows Help
# 02_12
from time import *
from functools import wraps

def pamti(funk):
    izračunano = {}
    @wraps(funk)
    def wrap(*args):
        if args not in izračunano:
            izračunano[args] = funk(*args)
        return izračunano[args]
    return wrap

@pamti
def f1(n):
    if n < 4:
        return n
    else:
        return f1(n - 1) + f1(n - 2) + f1(n - 3)


def f2(n):
    if n < 4:
        return n
    else:
        return f2(n - 1) + f2(n - 2) + f2(n - 3)

n = int(input())
poc = clock()
f1(n)
kraj = clock()
print('Trajanje funkcije s pamćenjem: {0}ms'.format((kraj - poc) * 1000))
poc = clock()
f2(n)
kraj = clock()
print('Trajanje obične rekurzije: {0}ms'.format((kraj - poc) * 1000))
```