Cvicenie 5

Instrukcie:

- Vypracujte vsetky ulohy. Na cviceni sa pokuste vypracovat co najviac uloh a ulohy, ktore nestihnete na cviceni, potom vypracujte doma.
- **Pozor! Nevytvarajte rekurzivne funkcie pomocou globalnych premennych!** Je to zly zvyk a na skuske bude za taketo riesenia 0 bodov!
- **Specialne davam do pozornosti ulohu cislo 7, v ktorej sa oboznamite so slavnou a podla mna aj velmi peknou algoritmickou ulohou „Towers of Hanoi“.**
- **V pripade, ze sa na niektorej ulohe zaseknete, pytajte sa cviciaceho.**

Cast prva: Debugger

1. Vyvojove prostredia pre tvorbu pocitacovych programov spravidla obsahuju nastroj zvany Debugger. Pomocou tohto nastroja mozete prechadzat vasim programom krok po kroku a mozete sledovat, ako sa menia hodnoty premennych vo vasom programe. Toto je velmi uzitocne pri hladani chyb vo vasom programe. Prejdite si tutorial o pouzivani Debuggera vo vyvojovom prostredi IDLE dostupny na:

https://www.cs.uky.edu/~keen/help/debug-tutorial/debug.html

Osvojte si pracu s Debuggerom v IDLE. **Bude to pre vas uzitocne!** Ak radsej pouzivate ine vyvojove prostredie ako IDLE, osvojte si pracu s Debuggerom vo vasom preferovanom vyvojovom prostredi!

Cast druha: Rekurzia

1. Definujte funkciu s parametrom n, ktora vrati sucet 1+2+3+...+n. Vo funkcii pouzite **rekurziu**!
Priklad cinnosti funkcie: funkcia pre hodnotu parametra n=1 **vrati** 1, funkcia pre hodnotu parametra n=5 **vrati** 15, pre hodnotu parametra n=10 **vrati** 55.

2. Definujte funkciu s parametrom n. Funkcia nacita z klavesnice n cisel a vrati sucet nacitanych cisel. Vo funkcii pouzite **rekurziu**!
Priklad cinnosti funkcie: funkcia pre hodnotu parametra n=5 nacita 5 cisiel. Ak budu nacitane cisla 7, 4, -2, 4, -7, funkcia **vrati** cislo 6.

3. Definujte funkciu s parametrom n. Funkcia nacita z klavesnice n cisel a vrati pocet, kolko z nacitanych cisel bolo parnych. Vo funkcii pouzite **rekurziu**!
Priklad cinnosti funkcie: funkcia pre hodnotu parametra n=5 nacita 5 cisiel. Ak budu nacitane cisla 7, 4, 0, 10, -7, funkcia **vrati** cislo 3 (lebo 4, 0 a 10 su parne). Ak budu nacitane cisla 7, 3, 1, 11, -7, funkcia **vrati** cislo 0 (lebo ziadne z 5 nacitanych cisiel nebolo parne).

4. Definujte funkciu s parametrom n. Funkcia nacita z klavesnice n cisel a vrati najvacsie z nacitanych cisel. Vo funkcii pouzite **rekurziu**!
Priklad cinnosti funkcie: funkcia pre hodnotu parametra n=5 nacita 5 cisiel. Ak budu nacitane cisla 7, 4, 0, 10, -7, funkcia **vrati** cislo 10.

5. Definujte funkciu s parametrom n. Funkcia nacita z klavesnice n cisel a vrati hodnotu True, ak je sucet nacitanych cisel parny. V opacnom pripade vrati funkcia hodnotu False. Vo funkcii pouzite **rekurziu**!
Pomocka: Zamyslite sa, ako sa zmeni parita cisla, ak k nemu pridame parne cislo, a ako sa zmeni, ak k nemu pridame neparne cislo.

Priklad cinnosti funkcie: funkcia pre hodnotu parametra n=5 nacita 5 cisiel. Ak budu nacitane cisla 7, 4, 0, 10, -7, funkcia **vrati** True, pretoze 7+4+0+10+(-7) = 14, co je parne cislo.

6. Definujte funkciu, ktora pre argument n vrati sucet prvocisel mensich ako n. Vo funkcii pouzite **rekurziu**! Mozete predpokladat, ze mate k dispozicii funkciu *test_prvociselnosti* z minuleho cvicenia.
Priklad cinnosti funkcie: funkcia pre hodnotu parametra n=10 **vrati** cislo 17, pretoze prvocisla mensie ako 10 su 2, 3, 5, 7 a 2+3+5+7 = 17. Funkcia pre hodnotu parametra n=17 **vrati** cislo 41, pretoze prvocisla mensie ako 17 su 2, 3, 5, 7, 11, 13 a 2+3+5+7+11+13 = 41.

7. Vsetky vyssie uvedene ulohy by sa dali jednoducho vyriesit aj bez rekurzie. Rekurzia nam ale niekedy umoznuje jednoducho riesit aj ulohy, ktore by sa nam inak riesili velmi tazko. Prikladom takejto ulohy je slavna algoritmicka uloha “Towers of Hanoi”.

a) Pozrite si toto video o ulohe “Towers of Hanoi”:

https://www.youtube.com/watch?v=8lhxIOAfDss

Podla mna je velmi pekne a zaujimave :)

Komentar1:
Vo videu pouziva profesor Altenkirch prostredie Jupyter. Rovnake funkcie, ktore vytvara a spusta prof Altenkirch vo videu, si ale mozete vytvorit a spustit aj v prostredi Idle.

Komentar2:
Vo videu vytvori prof Altenkirch funkciu

```
def move(f,t):
    print("Move a disc from {} to {}!".format(f,t))
```

V tejto funkcii vyuziva prof Altenkirch metodu format() , s ktorou sme sa este na predmete nestretli. Ak by sme sa chceli vyhnut metode format(), mohli by sme funkciu move definovat nasledovne

```
def move(f,t):
    print("Move a disc from "+f+" to "+t+"!")
```

Takto definovana funkcia move ma rovnaky efekt ako funkcia move od prof Altenkircha. Namiesto metody format() sa v nej vyuziva iba spajanie retazcov, ktore sme preberali este na prvej prednaske.

b) V skripte vo videu vyssie je funkcia hanoi(n, f, h, t), ktora vypise sadu instrukcii ako vyriesit ulohu “Towers of Hanoi”, ak na zaciatku mame n diskov na stlpe s nazvom f a tieto disky chceme premiestnit na stlp s nazvom t a mame este k dispozicii stlp s nazvom h. Vasou ulohou je teraz definovat funkciu hanoi_count(n), ktora vrati, kolko presunov diskov treba vykonat, ak chceme vyriesit ulohu “Towers of Hanoi” s n diskami (inak povedane funkcia hanoi_count(n) vrati pocet riadkov, ktore vypise funkcia hanoi(n, f, h, t)). Funkcia hanoi_count(n) musi byt rekurzivna, nesmie volat funkciu hanoi(n, f, h, t) a nesmie nic vypisovat.

8. Vyrieste cvicenie 6.3 na strane 61 v knihe.
Priklad cinnosti funkcie, ktoru mate v danom cviceni vytvorit: volanie *is_palindrome(“anna”)* **vrati** True, pretoze retazec “anna” je palindrom. Volanie *is_palindrome(“anno”)* **vrati** False, pretoze retazec “anno” nie je palindrom.

9. Vyrieste cvicenie 6.4 na strane 61 v knihe.
Priklad cinnosti funkcie, ktoru mate v danom cviceni vytvorit: volanie *is_power(8,2)* **vrati** True, pretoze $8 = 2^3$. Volanie *is_power(1,3)* **vrati** True, pretoze $1 = 3^0$. Volanie *is_power(81,3)* **vrati** True, pretoze $81 = 3^4$. Volanie *is_power(80,3)* **vrati** False, pretoze 80 nie je mocnina 3.

10. Vyrieste cvicenie 6.5 na strane 61 v knihe.
Priklad cinnosti funkcie, ktoru mate v danom cviceni vytvorit: volanie gcd*(10,15)* **vrati** 5, volanie gcd*(30,40)* **vrati** 10, volanie gcd*(20,27)* **vrati** 1**.**

11. Vyrieste cvicenie 5.6 na strane 49 v knihe.

12. Definujte funkciu, ktora pre argument *n ,* ktorym je nezaporne cele cislo, **vypise na obrazovku** binarny rozvoj cisla *n.* Vo funkcii pouzite **rekurziu**!
Priklad cinnosti funkcie: funkcia pre hodnotu parametra n=10 **vypise** na obrazovku 1010.
Pre parameter n = 0 **vypise** na obrazovku 0. Pre parameter n = 31 **vypise** na obrazovku 11111.
(Tato uloha je inspirovana ulohou c. 175 z knihy Python Workbook od Bena Stephensona).

13. Definujte funkciu pracujucu **rekurzivne**, ktora zisti (vrati True / False), ci je mozne vytvorit sumu v EURACH pomocou zadaneho poctu minci. Uvazujme len centove mince, t.j. 0.01€, 0.02€, 0.05€, 0.10€, 0.20€, 0.50€. V ramci zadaneho poctu minci sa mozu uvedene mince aj opakovat. Parametre funkcie **musia obsahovat** celkovu sumu a pocet minci, avsak ak to uznate za vhodne, funkciu mozete vytvorit aj s dalsimi parametrami.
Priklad cinnosti funkcie:
Suma 0.04€ pomocou 4 minci – True, pretoze 0.04€ je mozne zostrojit pomocou 4 minci, kazda v hodnote 0.01€.
Suma 0.60€ pomocou 2 minci – True, pretoze 0.60€ je mozne zostrojit ako 0.50€ + 0.10€.
Suma 0.31€ pomocou 2 minci – False, pretoze 0.31€ nie je mozne zostrojit pomocou 2 minci ziadnym sposobom.
Suma 0.31€ pomocou 3 minci – True, pretoze 0.31€ je mozne zostrojit ako 0.20€+0.10€+0.01€.
(Tato uloha je inspirovana ulohou c. 181 z knihy Python Workbook od Bena Stephensona).

13b. Upravte predoslu funkciu tak, aby funkcia okrem vratenia True/False v pripade, ze je sumu mozne zostrojit, vypisala, ake mince je potrebne pouzit.