Úvodné poznámky

Všetky vzorové riešenia sú len **vzorové**, t. j. nie sú to jediné možné riešenia daných úloh a ani si nenárokujú byť najkrajšie. V matematike, informatike a kryptológii je len veľmi málo takých úloh, ktoré by sa dali riešiť len jediným možným spôsobom. K riešeniu úloh spravidla vedie viacero ciest a je len na riešiteľovi, či ním nájdená cesta bude jednoduchšia alebo zložitejšia, kratšia alebo dlhšia, krajšia alebo menej pekná. Podstatné je, aby zvolená cesta k riešeniu bola správna, čiže aby použité metódy boli vhodné a boli aj správne použité.

- **Matematika** – Pri riešení úloh v centaurónskej matematike využívame len vedomosti známe väčšinou už zo základnej školy. Hlavne ide o fakt, že každú rovnicu $a \cdot x \,(\mathrm{mod}\, b) = c$ vieme zapísať v tvare

  $$a \cdot x = b \cdot k + c\,, \quad \text{kde} \quad x \in \{0, 1, \ldots, (b-1)\} \quad \text{a} \quad k \in \mathbb{Z}\,.$$

  Okrem toho v týchto úllohách ešte využívame *Základnú vetu aritmetiky*, ktorá hovorí o tom, že každé prirodzené (celé) číslo sa dá jednoznačne (jediným spôsobom) zapísať ako súčin mocnín prvočísel.

- **Informatika** – V informatických úlohách sa môžu riešenia líšiť nielen samotným kódom programu (iné mená premenných, iné komentáre, atď), ale aj použitými metódami riešenia. V úlohe 5A napríklad nemusíte nič vedieť o *Zellerovom algoritme*, ktorý sa spomína vo vzorovom riešení. Úloha sa dá riešiť aj oveľa triviálnejším, aj keď oveľa menej elegantným spôsobom.

- **Kryptológia** – Kryptografické úlohy (obvykle lúštenie šifier) sú spravidla na riešenie zložitejšie než matematické alebo informatické úlohy. Vyžadujú si veľa experimentovania, skúšania mnohých možností, ktoré sa potom ukážu byť slepými cestami, až napokon dostanete spásonosný nápad a nájdete správne riešenie. Preto tu uvedené vzorové riešenia týchto úloh neberte ako „kompletné“ riešenia, ale len ako ten finálny nápad, ktorý viedol k cieľu. Tomu však muselo predchádzať množstvo neúspešných pokusov a vylučovanie nesprávnych možností.

5 bodov

1A. V centaurónskej aritmetike nájdite všetky riešenia rovnice $17 \cdot x = 17$.

Riešenie: Rovnicu $17 \cdot x = 17$ v centaurónskej aritmetike si môžeme v našej (modulárnej) aritmetike prepísať do podoby

$$17 \cdot x \,(\mathrm{mod}\, 100) = 17$$

čo je to isté, ako keby sme napísali

$$17x = 100k + 17\,, \quad \text{kde} \quad x \in \{0, 1, \ldots, 99\} \quad \text{a} \quad k \in \mathbb{Z}\,.$$

Poslednú rovnicu môžeme ďalej upraviť

$$\begin{aligned} 17x &= 100k + 17 \\ 17(x-1) &= 100k \\ 17(x-1) &= 2^2 5^2 k \end{aligned}$$

Čísla $2, 5$ a 17 sú prvočísla, a preto podľa základnej vety aritmetiky má posledná rovnica riešenia práve vtedy, ak je $(x-1)$ celočíselným násobkom $2^2 5^2$ a súčasne k je vhodným násobkom čísla 17. A to je pre $x \in \{0, 1, \ldots, 99\}$ možné len v prípade ak

$$(x-1) = k = 0 \quad \Rightarrow \quad x = 1\,.$$

Preto jediné riešenie danej rovnice je $x = 1$.

5 bodov ♣

1B. Dokážte, že v centaurónskej aritmetike má rovnica $a \cdot x = a$ pre ľubovoľné a nesúdeliteľné s číslami 2 a 5, vždy len jedno riešenie.

Riešenie: Predovšetkým si musíme uvedomiť, že $x = 1$ bude riešením každej rovnice $a \cdot x = a$ v centaurónskej, alebo modulárnej aritmetike. Potrebujeme preto dokázať, že ak je a nesúdeliteľné s číslami 2 a 5, tak iné riešenie neexistuje.
Dôkaz budeme robiť sporom. Predpokladajme, že existujú dve rôzne riešenia danej rovnice a označme si tieto riešenia x_1 a x_2. Riešenia sú rôzne, a preto $x_1 \neq x_2$. Potom platí

$$\begin{aligned} ax_1 &= 100k + a \\ ax_2 &= 100l + a\,, \quad \text{kde} \quad x_1, x_2 \in \{0, 1, \ldots, 99\} \quad \text{a} \quad k, l \in \mathbb{Z}\,. \end{aligned}$$

Ak od prvej rovnice odčítame druhú rovnicu, tak dostaneme

$$a(x_1 - x_1) = 100(k - l)\,.$$

Poslená rovnica má, rovnako ako posledná rovnica v riešení úlohy 1A, riešenie len v prípade ak

$$(x_1 - x_2) = (k - l) = 0 \quad \Rightarrow \quad x_1 = x_2\,.$$

Dostali sme teda rovnosť $x_1 = x_2$, čo je spor s predpokladom, že $x_1 \neq x_2$. Pôvodný predpoklad bol preto nesprávny a platí jeho negácia, t. j. neexistujú dve rôzne riešenia danej rovnice ak a je nesúdeliteľné s číslami 2 a 5 a jej jediné riešenie je v takom prípade $x = 1$.

7 bodov ♢

2A. V centaurónskej aritmetike nájdite riešenie rovnice $23 \cdot x = 1$.

Riešenie: Riešenie rovnice $23 \cdot x = 1$ v centaurónskej aritmetike sa dá nájsť pomocou *rozšíreného Euklidovho algoritmu*. To tu však nebudeme robiť. Daná rovnica sa dá zapísať v tvare

$$23x = 100k + 1\,, \quad \text{kde} \quad x \in \{0, 1, \ldots, 99\} \quad \text{a} \quad k \in \mathbb{Z}\,.$$

Číslo na pravej strane rovnice končí cifrou 1. Číslo 23 končí cifrou 3, a preto musí číslo x končiť cifrou 7. Z uvedeného vieme, že x môže byť len z množiny $x \in \{1, 17, 27, 37, 47, 57, 67, 77, 87, 97\}$. Stačí nám teda vyskúšať uvedených 10 čísel a nájsť riešenie, ktorým je číslo 87.

---

7 bodov ♢

2B. V centaurónskej aritmetike nájdite všetky riešenia rovnice $34 \cdot x = 1$.

Riešenie: Túto rovnicu by sme opäť mohli riešiť rozšíreným Euklidovým algoritmom, čo nebudeme robiť. Zistili by sme, že nemá riešenie. To však vieme zistiť aj inak. Danú rovnicu si zapíšeme v tvare

$$34x = 100k + 1\,, \quad \text{kde} \quad x \in \{0, 1, \ldots, 99\} \quad \text{a} \quad k \in \mathbb{Z}\,.$$

Na ľavej strane uvedenej rovnice máme $34x$, čo bude vždy párne číslo a na pravej strane rovnice máme $100k + 1$, čo bude vždy nepárne číslo. Z toho je zrejmé, že daná rovnica nebude mať riešenie.

8 bodov ♡

3A. V centaurónskej aritmetike nájdite všetky riešenia rovnice $x^4 = 16$.

Riešenie: Rovnako ako v predchádzajúcich úlohach, aj tu si rovnicu $x^4 = 16$ z centaurónskej aritmetiky, vieme prepísať do tvaru

$$x^4 = 100k + 16\,, \quad \text{kde} \quad x \in \{0, 1, \ldots, 99\} \quad \text{a} \quad k \in \mathbb{Z}\,.$$

Z toho vidíme, že x^4 musí končiť dvojčíslím 16, resp. cifrou 6. Vieme, že $x^4 = x^2 \cdot x^2 = (x^2)^2$.

- ▷ Cifru 6 sa x^2 môže končiť len vtedy, ak sa x končí cifrou 4 alebo cifrou 6.
- ▷ Cifru 6 sa x^4 môže končiť len vtedy, ak sa x končí cifrou 2, cifrou 4 alebo cifrou 6.

Pre číslo x máme preto len 30 možností $x \in \{2, 4, 6, 12, 14, 16, \ldots, 92, 94, 96\}$, ktoré stačí vyskúšať a nájdeme všetky riešenia danej rovnice. Tými sú $x \in \{2, 14, 36, 48, 52\}$.

---

8 bodov ♡

3B. V centaurónskej aritmetike nájdite všetky riešenia rovnice $x^3 = 14$.

Riešenie: Danú rovnicu, rovnako ako v predošlej úlohe, prepíšeme do tvaru

$$x^3 = 100k + 14\,, \quad \text{kde} \quad x \in \{0, 1, \ldots, 99\} \quad \text{a} \quad k \in \mathbb{Z}\,.$$

Z uvedeného vidíme, že x^3 sa bude končiť dvojčíslím 14, resp. cifrou 4.
Pre žiadne z čísel $x \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ sa x^3 nekončí cifrou 4. Z toho je zrejmé, že rovnica $x^3 = 14$ v centaurónskej aritmetike nebude mať riešenie.

10 bodov

4A. V centaurónskej aritmetike nájdite všetky riešenia rovnice $13^x = 17$.

Riešenie: Uvedená úloha úzko súvisí so šifrovacím algoritmom RSA a existuje jednoduchý spôsob jej riešenia pomocou *rozšíreného Euklidovho algoritmu*. Na rozdiel od šifry RSA počíta centaurónska aritmetika všetko $(\mathrm{mod}\, 100)$ a vieme, že $100 = 2^2 5^2$. Preto máme všetky potrebné informácie na riešenie úlohy pomocou dešifrovacieho algoritmu pre RSA. Toto však nechceme robiť, pretože nepredpokladáme, že by stredoškoláci poznali šifru RSA, alebo vedeli používať rozšírený Euklidov algoritmus na hľadanie inverzných prvkov v modulárnej aritmetike.
Preto danú rovnicu prepíšeme do tvaru

$$13^x = 100k + 17\,, \quad \text{kde} \quad x \in \{0, 1, \ldots, 99\} \quad \text{a} \quad k \in \mathbb{Z}\,.$$

Celočíselné mocniny čísla, ktoré sa končí cifrou 3, sa končia iba ciframi 3, 9, 7 a 1 v uvedenom poradí a tieto štyri cifry sa cyklicky opakujú. Z toho vieme, že $x\,(\mathrm{mod}\, 4) = 3$ inak povedané, číslo x musí byť v tvare $x = 4k + 3$, kde $k \in \mathbb{Z}$. To nám počet možností, ktoré treba vyskúšať obmedzí na 25

$$x \in \{3, 7, 11, \ldots, 95, 99\}\,.$$

Vyskúšaním týchto možností zistíme, že rovnica má riešenia

$$x \in \{7, 27, 47, 67, 87\}\,.$$

10 bodov

4B. V centaurónskej aritmetike nájdite všetky riešenia rovnice $36^x = 96$.

Riešenie: Rovnako ako v úlohe 4A si danú rovnicu prepíšeme do tvaru

$$36^x = 100k + 96\,, \quad \text{kde} \quad x \in \{0, 1, \ldots, 99\} \quad \text{a} \quad k \in \mathbb{Z}\,.$$

V tomto prípade nám nestačí skúmať poslednú cifru, pretože každá celočíselná mocnina čísla, ktoré sa končí cifrou 6, sa tiež končí cifrou 6. V tomto prípade musíme skúmať posledné dve cifry. Napíšeme si tabuľku mocnín čísla 36 a posledných dvojčíslí, ktorými sa tieto mocniny končia.

n	36^n
1	...36
2	...96
3	...56
4	...16
5	...76
6	...36

Z uvedenej tabuľky vidíme, že dvojčíslím 96 sa končia všetky mocniny 36^x, kde x je v tvare $x = 5k+2$. Riešení danej rovnice je preto 20 a súnimi čísla

$$x \in \{2, 7, 12, 17, 22, 27, 32, 37, 42, 47, 52, 57, 62, 67, 72, 77, 82, 87, 92, 97\}\,.$$

6 bodov ♡

5A. V Pythone naprogramujte „nekonečný“ kalendár. Vstupom programu bude ľubovoľný platný dátum v tvare `DD.MM.YYYY` (vždy 10 znakov). Úlohou je určiť názov dňa v týždni, na ktorý tento dátum pripadá (t. j. pondelok, utorok, atď.).

- **Vstup:**
  - ◇ Dátum vo formáte `DD.MM.YYYY`.
  - ◇ Rok bude zadaný v intervale 1583 – 9999.
- **Výstup:**
  - ◇ Buď názov dňa v týždni (t. j. pondelok, utorok, atď.),
  - ◇ alebo poradové číslo dňa v týždni

    1 = pondelok
    2 = utorok
    3 = streda
    4 = štvrtok
    5 = piatok
    6 = sobota
    7 = nedeľa
- **Príklady:**
  - ◇ Vstup: 18.07.2049 → Výstup: nedeľa (alebo 7)
  - ◇ Vstup: 01.01.7845 → Výstup: streda (alebo 3)
  - ◇ Vstup: 29.02.3008 → Výstup: pondelok (alebo 1)

Riešenie: Vytvorenie „nekonečného kalendára“ je programátorská úloha, ktorá preverí pochopenie práce s dátumami a algoritmami. Keďže sa pohybujeme v rokoch 1583 až 9999, používame Gregoriánsky kalendár. Najefektívnejším spôsobom, ako vyriešiť túto úlohu bez špecializovaných externých knižníc, je použiť Zellerov algoritmus.

Zellerov algoritmus

Zellerov algoritmus premieňa kalendárne údaje na čisto matematický problém. Namiesto prácneho sčítania dní v roku používa jeden súhrnný vzorec. Tu je princíp Zellerovho algoritmu zhrnutý v 3 bodoch:

1. **Úprava mesiacov** (kľúčový trik)

   Zeller si všimol, že dĺžky mesiacov (30, 31, 28/29 dní) spôsobujú nepravidelnosť. Vyriešil to tak, že január a február presunul na koniec predchádzajúceho roka (ako 13. a 14. mesiac).

   - Vďaka tomu rok vždy začína marcom.
   - Marec má 31 dní, apríl 30, máj 31… táto postupnosť je matematicky predvídateľnejšia.

- Priestupný deň (29. február) sa tak ocitne až na úplnom konci výpočtu, kde nespôsobuje chyby v predchádzajúcich mesiacoch.

2. **Matematický vzorec**

   Algoritmus používa nasledovný vzorec:

$$h = \left(q + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \left\lfloor \frac{K}{4} \right\rfloor + \left\lfloor \frac{J}{4} \right\rfloor - 2J \right) \bmod 7$$

   Vysvetlivky:

   - h je deň v týždni: 0 = sobota, 1 = nedeľa, 2 = pondelok, 3 = utorok, 4 = streda, 5 = štvrtok a 6 = piatok,
   - q je deň v mesiaci,
   - m je mesiac: 3 = marec, 4 = apríl, 5 = máj,..., 14 = február,
   - K predstavuje posledné dve číslice roku (vypočítame ho teda ako $rok \bmod 100$), pre január a február je o 1 nižší (pretože sa počítajú do predchádzajúceho roku),
   - J predstavuje prvé dve číslice roku (vypočítame ako $\lfloor rok/100 \rfloor$), napríklad pre rok 1956 je $J = 19$.

   Poznámka:
   V tomto algoritme zápis `DD.MM.YYYY` zodpovedá zápisu `q.m.JK`, pričom január a február sú kvôli priestupným rokom počítané ako 13. a 14. mesiac predchádzajúceho roku. Napríklad dátum `02.02.2010` algoritmus počíta ako 2. deň 14. mesiaca roku 2009.

   Vyššie uvedený vzorec v podstate sčítava štyri vplyvy na posun dňa v týždni:

   - Deň v mesiaci (q): Každý deň prirodzene posúva výsledok o 1.
   - Mesačná zložka: Výraz $\lfloor \frac{13(m+1)}{5} \rfloor$ je „magická časť“, ktorá vďaka posunu začiatku roka na marec presne simuluje striedanie 30 a 31-dňových mesiacov.
   - Ročná zložka (K): Zohľadňuje posun o jeden deň každý rok (365 dní mod 7 = 1).
   - Zložka priestupných rokov: Časti $\lfloor \frac{K}{4} \rfloor$ a $\lfloor \frac{J}{4} \rfloor - 2J$ pridávajú extra dni za každé 4 roky, ale zároveň odčítavajú korekcie za celé storočia (keďže roky 1700, 1800, 1900 neboli priestupné).

3. **Výsledok** (modulo 7)

   Celý súčet sa vydelí číslom 7 a zaujíma nás len zvyšok po delení. Tento zvyšok (0 až 6) priamo zodpovedá konkrétnemu dňu v týždni, pričom v zmysle Zellerovho algoritmu platí 0 = sobota, 1 = nedeľa, 2 = pondelok, 3 = utorok, 4 = streda, 5 = štvrtok a 6 = piatok. Vzhľadom na zadanie úlohy je potrebné si prispôsobiť toto mapovanie tak, aby týždeň začínal pondelkom, t. j. 0 = pondelok, 1 = utorok, atď.

Príklady

- **Príklad 1**. Vstupný dátum: `01.01.2000`. Podľa algoritmu, január chápeme ako 13. mesiac roku 1999. Pre výpočet dosadíme tieto hodnoty:

  $q = 1$ $m = 13$ $K = 99$ $J = 19$

  Výsledná hodnota vypočítaná vzorcom je $(1 + 36 + 99 + 24 + 4 - 38) \bmod 7 = 126 \bmod 7 = 0$ (sobota). Avšak v zmysle zadania má sobota poradové číslo 6. Ak program vypíše 6 alebo reťazec `sobota`, obidve verzie sú považované za správne.

- **Príklad 2**. Vstupný dátum: `08.09.1944`. Pre výpočet dosadíme tieto hodnoty:

  $q = 8$ $m = 9$ $K = 44$ $J = 19$

  Výsledná hodnota vypočítaná vzorcom je $(1 + 26 + 44 + 11 + 4 - 38) \bmod 7 = 48 \bmod 7 = 6$ (piatok). Avšak v zmysle zadania má piatok poradové číslo 5. Ak program vypíše 5 alebo reťazec `piatok`, obidve verzie sú považované za správne.

Implementácia v Pythone

Príklad zdrojového kódu, ktorý spracuje vstupný reťazec, aplikuje matematický vzorec a vypíše výsledok, je na ďalšej strane.

Príklady výstupov pre jednotlivé vstupné dátumy.

```
Vstup: 18.07.2049 -> Výstup: nedeľa (7)
Vstup: 01.01.7845 -> Výstup: streda (3)
Vstup: 29.02.3008 -> Výstup: pondelok (1)
```

Zdrojový kód:

```
def urci_den_v_tyzdni(datum_str):
    # 1. Rozdelenie vstupu na deň, mesiac a rok
    den = int(datum_str[0:2])
    mesiac = int(datum_str[3:5])
    rok = int(datum_str[6:10])

    # 2. Úprava pre Zellerov algoritmus
    # V tomto algoritme sa január a február počítajú ako 13. a 14.
    # mesiac predchádzajúceho roka
    if mesiac < 3:
        mesiac += 12
        rok -= 1

    K = rok % 100    # Rok v storočí
    J = rok // 100   # Storočie

    # 3. Zellerov vzorec pre Gregoriánsky kalendár
    # Výsledok 'h' je deň v týždni (0 = sobota, 1 = nedeľa, ..., 6 = piatok)
    h = (den + ((13 * (mesiac + 1)) // 5) + K + (K // 4) + (J // 4) - (2 * J)) % 7

    # 4. Prevod výsledku na slovenské názvy a poradové čísla (1=pondelok ... 7=nedeľa)
    # Zeller: 0=So, 1=Ne, 2=Po, 3=Ut, 4=St, 5=Št, 6=Pi
    mapovanie = {
        2: ("pondelok", 1),
        3: ("utorok", 2),
        4: ("streda", 3),
        5: ("štvrtok", 4),
        6: ("piatok", 5),
        0: ("sobota", 6),
        1: ("nedeľa", 7)
    }

    nazov, poradie = mapovanie[h]
    return f"{nazov} ({poradie})"

# Zadanie vstupného dátumu

datum = input()

# Vypocet a vypis
print(urci_den_v_tyzdni(datum))
```

6 bodov ♡

5B. V Pythone napíšte program, ktorý vypočíta počet dní medzi dvoma platnými dátumami v tvare `DD.MM.YYYY` (vždy 10 znakov). Prvý a druhý zadaný dátum sa do počtu dní nerátajú a nezáleží na poradí, v ktorom dni zadáme.

- **Vstupy:**
  - Dva dátumy vo formáte `DD.MM.YYYY`.
  - Rok bude zadaný v intervale 1583 – 9999.
- **Výstup:**
  - Číslo udávajúce počet dní medzi zadanými dátumami.
- **Príklady:**
  - Vstupy: 01.01.2026 , 06.01.2026 → Výstup: 4
  - Vstupy: 24.12.2025 , 06.12.2025 → Výstup: 17

Riešenie: Najefektívnejší spôsob, ako vypočítať rozdiel medzi dvoma dátumami, je previesť každý z nich na celkový počet dní od nejakého fixného bodu v minulosti (napr. od roku 0). Rozdiel týchto dvoch hodnôt nám potom dá presný počet dní medzi nimi. Je pravdou, že historicky Gregoriánsky kalendár v roku 0 neexistoval. Avšak z hľadiska algoritmického riešenia úloh je to úplne v poriadku. V programovaní používame termín proleptický gregoriánsky kalendár. To znamená, že pravidlá nášho súčasného kalendára aplikujeme spätne v čase, aj na obdobie, kedy ešte neplatili.

Implementácia v Pythone

Príklad zdrojového kódu v Pythone, ktorý vypočíta počet dní medzi dvoma platnými dátumami, pričom obidva zadané dátumy sa do počtu dní nerátajú, je uvedený na nasledujúcej strane.

Príklady výstupov pre jednotlivé vstupné dátumy:

```
Zadaj prvý dátum (DD.MM.YYYY): 01.01.2026
Zadaj druhý dátum (DD.MM.YYYY): 06.01.2026
Výstup: 4

Zadaj prvý dátum (DD.MM.YYYY): 24.12.2025
Zadaj druhý dátum (DD.MM.YYYY): 06.12.2025
Výstup: 17
```

```
def je_priestupny(rok):
    """Vráti True, ak je rok priestupný, inak False."""
    # Rok je priestupný ak je deliteľný 4,
    # ale nie 100, pokiaľ nie je zároveň deliteľný 400.
    return (rok % 4 == 0 and rok % 100 != 0) or (rok % 400 == 0)

def dni_v_mesiaci(mesiac, rok):
    """Vráti počet dní v danom mesiaci a roku."""
    if mesiac in [4, 6, 9, 11]:
        return 30
    if mesiac == 2:
        return 29 if je_priestupny(rok) else 28
    return 31

def datum_na_dni(datum_str):
    """Prevedie reťazec DD.MM.YYYY na celkový počet dní od roku 0."""
    den = int(datum_str[0:2])
    mesiac = int(datum_str[3:5])
    rok = int(datum_str[6:10])

    celkovo_dni = 0

    # 1. Pripočítame dni za celé roky
    for r in range(1, rok):
        celkovo_dni += 366 if je_priestupny(r) else 365

    # 2. Pripočítame dni za celé mesiace v aktuálnom roku
    for m in range(1, mesiac):
        celkovo_dni += dni_v_mesiaci(m, rok)

    # 3. Pripočítame dni v aktuálnom mesiaci
    celkovo_dni += den

    return celkovo_dni

def vypocitaj_rozdiel():
    # Vstup od používateľa
    vstup1 = input("Zadaj prvý dátum (DD.MM.YYYY): ")
    vstup2 = input("Zadaj druhý dátum (DD.MM.YYYY): ")

    # Prevod dátumov na celkové počty dní
    dni1 = datum_na_dni(vstup1)
    dni2 = datum_na_dni(vstup2)

    # Absolútny rozdiel dní
    rozdiel = abs(dni1 - dni2)
```

... pokračovanie na ďalšej strane

```
    # Úloha vyžaduje počet dní "medzi", čiže krajné dni sa nerátajú.
    # Ak sú dni rovnaké, rozdiel je 0. Ak sú rôzne (napr. 1. a 3.),
    # matematický rozdiel je 2, ale "medzi" nimi je len 1 deň.
    # Preto musíme odčítať 1, ak sú dátumy odlišné.
    if rozdiel > 0:
        vysledok = rozdiel - 1
    else:
        vysledok = 0

    print(f"Výstup: {vysledok}")

# Spustenie programu
if __name__ == "__main__":
    vypocitaj_rozdiel()
```

Poznámky k implementácii

- Priestupný rok: pravidlo pre Gregoriánsky kalendár hovorí, že rok je priestupný, ak je deliteľný 4. Ak je však deliteľný 100, musí byť deliteľný aj 400, aby bol priestupný (preto rok 1900 nebol, ale rok 2000 bol priestupný).
- Normalizácia na dni: Funkcia `datum_na_dni` vypočíta, koľko dní uplynulo od začiatku nášho letopočtu až po zadaný dátum. Tým sa vyhneme zložitému počítaniu rozdielov medzi mesiacmi priamo.
- Výpočet „medzi“: Ak zadáme `01.01.2026` a `06.01.2026`, matematický rozdiel je $6 - 1 = 5$. Dni medzi nimi sú: 2., 3., 4. a 5. január (spolu 4 dni). Preto v kóde používame `rozdiel - 1`.

14 bodov

6A. Nájdite správu ukrytú v nasledujúcom obrázku.

Riešenie: Na obrázku sú rôzne objekty (domček, oblaky, slnko, vták, kvety, kamene, mravce,...). Avšak jediné objekty, ktoré sa opakujú dostatočne často, aby v nich mohla byť ukrytá správa, sú len kvety, kamene a mravce. Oblaky sa síce tiež opakujú, ale je ich príliš málo. Rovnako aj mravcov je na zmysluplnú správu príliš málo. Preto zostávajú len kvety a kamene, pričom kamene sú v rôznych veľkostiach a umiestnené príliš nepravidelne. Pozrime sa preto bližšie na kvety.

Kvety sa vyskytujú v 3 veľkostiach (druhoch): nízke, stredne vysoké a vysoké. Ak nízke kvety nahradíme symbolom ·, stredne vysoké symbolom − a vysoke symbolom | dostaneme

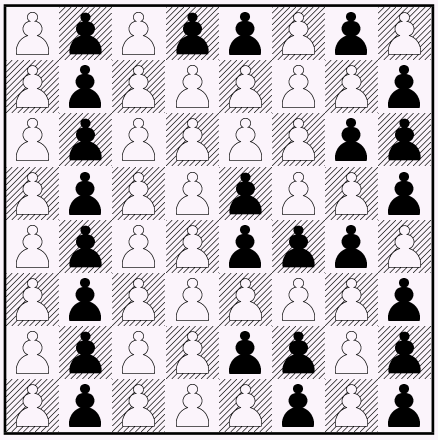
$$\cdot\,\cdot - \cdot \,|\, - - |\, \cdot\,\cdot |\, - \cdot\,\cdot -$$

Tento zápis nápadne pripomína Morseov kód. Pozrieme sa teda do tabuľky Morseovho kódu, odkiaľ zistíme, že ukrytá správa znie

FMIX

14 bodov

6B. Nájdite správu ukrytú v nasledujúcom obrázku.



Riešenie: Šachovnica má rozmer 8×8. Sú na nej bieli a čierni pešiaci, ktorí by mohli predstavovať 0 a 1. Najjednoduchší nápad je potom taký, že ukrytá správa by mohla byť zakódovaná binárne 8-bitovými číslami a mohla by pozostávať z 8 znakov.

Jednotlivé znaky by mohli byť buď riadky, alebo stĺpce šachovnice. Ak sa ale pozrieme na stĺpce šachovnice tak vidíme, že prvý stĺpec obsahuje len bielych a druhý stĺpec len čiernych pešiakov. To znamená samé 0, alebo samé 1, a preto sú stĺpce ako znaky vysoko nepravdepodobné. Venujme sa preto najskôr riadkom. Najznámejšie kódovanie znakov používané v počítačoch je ASCII kód. Štandardný ASCII je len 7-bitový kód, rozšírený ASCII je 8-bitový. Pritom písmená aj čísla sú všetky v rozsahu 0 až 127 (dekadický zápis), t. j. na ich kódovanie sa využíva len 7 najnižších bitov. Tomu by zodpovedal prvý stĺpec pozostávajúci zo samých 0, pretože 8. bit je vo všetkých písmenách a číslach 0. Prepíšeme si teda „šachovnicu“ do bitovej podoby a pozrime sa akým ASCII znakom zodpovedajú jej riadky

0101 1010	Z
0100 0001	A
0100 0011	C
0100 1001	I
0100 1110	N
0100 0001	A
0100 1101	M
0100 0101	E

Vidíme, že slovo na pravej strane dáva zmysel. V obrázku ukrytá správa je `ZACINAME`.