1) Jak bychom v typickém programovacím jazyce zapsali reálná čísla ve formátu fixed-point? Jaký datový typ bychom pro hodnotu použili v Pythonu?

2) Jaký typ z knihovny `numpy` bychom použili pro reprezentaci fixed-point čísel ve formátu 24.8? Pokud bychom chtěli uživateli vypsat obsah proměnné obsahující nějaké číslo v uvedeném formátu, tak jak bychom to v Pythonu zařídili, aby se se číslo vypsalo jako reálná hodnota v desítkové soustavě?

3) Je možné pro násobení fixed-point čísel použít celočíselné násobení? Pokud ano, tak jak?

4) **Pro lehce pokročilé** (je třeba trochu promyslet správný přístup, a připomenout si starší přednášky): Pokud provedeme pro větší hodnoty ve fixed-point 5.3 část příkladu z přednášky:

```
x = uint8(0b00100_010) # 4.25
y = uint8(0b00111_000) # 7
z = x * y
z = uint8(z >> 3)
print(z / (1 << 3))
```

Tak program zobrazí výslednou hodnotu `1,75` a nikoliv správnou `29,75`. Proč program tento výsledek vypíše, když je i hodnota `29,75` ve formátu fixed-point 5.3 přesně reprezentovatelná? Jak by se dal uvedený kód upravit, aby proměnné `x`, `y` na vstupu, a finální stav proměnné `z` stále obsahovaly hodnotu typu `uint8` ve formátu fixed-point 5.3, ale výsledek operace byla správná hodnota `29,75`.

5) Je možné pro dělení fixed-point čísel použít celočíselné dělení? Pokud ano, tak jak?

6) **Pro lehce pokročilé** (je třeba trochu promyslet správný přístup, a připomenout si starší přednášky): Pokud provedeme dělení ve fixed-point 5.3 jednoduchým přímočarým způsobem z přednášky:

```
x = uint8(0b00101_000) # 5
y = uint8(0b00011_000) # 3
z = x // y
z = uint8(z << 3)
print(z / (1 << 3))
```

tak program vypisuje velmi nepřesné hodnoty. Níže naleznete seznam příkladů vstupních hodnot (*1. sloupec*), a nesprávných výsledků, které program vrací (*3. sloupec*). Správné výsledky (*2. sloupec*) sice nejsou ve formátu fixed-point 5.3 přesně reprezentovatelné, nicméně pokuste se uvedený výpočet upravit tak, aby výsledná hodnota ve fixed-point 5.3 byla správné blíže (přesnější – viz *4. sloupec*) než vrací výpočet stávající (opět chceme, aby proměnné `x`, `y` na vstupu, a finální stav proměnné `z` stále obsahovaly hodnotu typu `uint8` ve formátu fixed-point 5.3).

výpočet	správný výsledek	vypisovaný výsledek	přesnější výsledek
5 / 3	1,66666…	1,0	1,625
7 / 3	2,33333…	2,0	2,25 (neočekávejte ještě přesnějších 2,375)
23 / 6,25	3,68	3,0	3,625
30 / 6,875	4,363636…	4,0	4,25 (neočekávejte ještě přesnějších 4,375)
30,5 / 5,875	5,1914893617…	5,0	5,125 (neočekávejte ještě přesnějších 5,25)

7) Zapište správné výsledky všech 5 výpočtů z otázky 6 (viz *2. sloupec*) jako číslo ve dvojkové soustavě v normalizovaném vědeckém formátu, pokud máme na mantisu k dispozici **pouze 8 bitů** (= přípravný krok pro reprezentaci hodnot ve floating-point reprezentaci). Samotný převod do dvojkové soustavy neprovádějte ručně (základní koncept reprezentace reálných čísel máme procvičený již z 9. sady SA úloh), ale použijte nějaký nástroj, např. web (pozor, nezapomeňte zde použít desetinnou tečku místo desetinné čárky):
[http://www.avrams.ro/Convert-real-numbers.html]
Je pro uvedené příklady tento zápis přesnější, než v otázce 6 použitá fixed-point 5.3 reprezentace reálných čísel (ve které bude každá z uvedených hodnot zabírat také 8 bitů)?

8) Zapište následující čísla jako znaménkovou hodnotu s posunem (bias):

a) 15 s posunem +127
b) -19 s posunem +127
c) -50 s posunem +2048
d) 0 s posunem +2048

9) Všechny níže uvedené hodnoty budeme ukládat do paměti na uvedené adresy jako reálná čísla ve floating-point reprezentaci (s pohyblivou řádovou čárkou) splňující pravidla IEEE 754. Napište hexdump (definice a vysvětlení viz zadání self-assessment úloh pro 7. přednášku) finální stavu paměti mezi adresami `$0010F300` až `$0010F32F` po uložení všech uvedených hodnot (předpokládejte, že původně paměť obsahuje samé byty s hodnotou `$00`). Hodnoty se ukládají v **Little Endian** pořadí. Nezapomeňte, že máte k dispozici slide z 10. přednášky. Samotný převod reálných čísel z desítkové do dvojkové soustavy neprovádějte ručně (základní koncept reprezentace reálných čísel máme procvičený již z 9. sady SA úloh), ale použijte nějaký vhodný nástroj – viz např. otázka 7 – zakódování reálného čísla ve dvojkové soustavě do uvedené reprezentace s pohyblivou řádovou čárkou ale již poveďte ručně:

a) na adresu `$0010F300` hodnotu `-3,5` v 8-bitovém formátu (z přednášky),
kde 2 bity s nejnižší váhou je mantisa se skrytou 1, následuje 5-bitový exponent (bias +15), a MSb je znaménkový bit
b) na adresu `$0010F302` hodnotu `118,33984375` v 16-bitovém formátu (IEEE 754 half precision),
kde 10 bitů s nejnižší váhou je mantisa se skrytou 1, následuje 5-bitový exponent (bias +15), a MSb je znaménkový bit,
c) na adresu `$0010F304` hodnotu `-118,33984375` ve 32-bitovém formátu (IEEE 754 single precision),
kde 23 bitů s nejnižší váhou je mantisa se skrytou 1, následuje 8-bitový exponent (bias +127), a MSb je znaménkový bit.
d) na adresu `$0010F310` hodnotu `-118,33984375` v 64-bitovém formátu (IEEE 754 double precision),
kde 52 bitů s nejnižší váhou je mantisa se skrytou 1, následuje 11-bitový exponent (bias +1023), a MSb je znaménkový bit.

Pro lehce pokročilé (speciální případy floating-point čísel, přesný mechanismus viz slide z přednášky je třeba aplikovat na konkrétní formát floating-point čísel):

e) na adresu `$0010F308` hodnotu +∞ (+nekonečno) ve 32-bitovém formátu (IEEE 754 single precision),
kde 23 bitů s nejnižší váhou je mantisa se skrytou 1, následuje 8-bitový exponent (bias +127), a MSb je znaménkový bit,
f) na adresu `$0010F30C` hodnotu `0,00000678` v 16-bitovém formátu (IEEE 754 half precision),
kde 10 bitů s nejnižší váhou je mantisa se skrytou 1, následuje 5-bitový exponent (bias +15), a MSb je znaménkový bit,
g) na adresu `$0010F30E` hodnotu `-0,0` v 16-bitovém formátu (IEEE 754 half precision),
kde 10 bitů s nejnižší váhou je mantisa se skrytou 1, následuje 5-bitový exponent (bias +15), a MSb je znaménkový bit,
h) na adresu `$0010F320` hodnotu `-3,25` v 80-bitovém formátu (Intel x86 extended precision),
kde 64 bitů s nejnižší váhou je mantisa **bez skryté 1** (tj. **všechny bity mantisy jsou vždy** v hodnotě uložené), následuje 15-bitový exponent (bias +16383), a MSb je znaménkový bit.

10) Jaké jsou výhody a nevýhody floating-point reprezentace reálných čísel oproti fixed-point reprezentaci?

11) Jakým způsobem bychom měli v programu srovnávat floating-point hodnoty na rovnost? Proč je takový způsob porovnání vhodný?

12) V jaké situaci nám může ve výpočtu ve floating-point reprezentaci vyjít výsledek *+nekonečno* nebo *-nekonečno*?

13) V jaké situaci nám může ve výpočtu ve floating-point reprezentaci vyjít výsledek *Not a Number*?

14) Mějme v proměnné A hodnotu NaN, a chceme zjistit zda výsledek nějakého výpočtu uložený v proměnné B je také NaN. Můžeme jednoduše srovnat proměnné A a B na rovnost, nebo zjištění zda B je NaN, musíme provést jiným způsobem? Proč?

15) Jaká je výhoda IEEE 754 aritmetiky, a v jaké situaci je výhodné, že operace z otázek 12 a 13 nejsou chybou ale platným výsledkem?

16) Co je tzv. *microcontroller* nebo *jednočipový počítač*?

17) Jaký je rozdíl mezi paměťmi ROM, PROM a EPROM?

18) Co je hlavní společná vlastnost pamětí ROM, PROM a EPROM? V čem se odlišují od pamětí SRAM a DRAM?

Poznámka: Cílem těchto úloh je, abyste si doma v rámci opakování látky z přednášky mohli sami ověřit, jak jste látce porozuměli. Úlohy jsou koncipovány víceméně přímočaře, a po projití poznámek a případně video záznamů z přednášek by jejich řešení mělo být zřejmé. Pro řešení úloh tedy není potřeba studium látky nad rámec probraný na přednáškách (nicméně je třeba mít i znalosti a pochopení látky z paralelně probíhajících přednášek a cvičení z předmětu Programování I). Pokud i po detailním a opakovaném projití látky z přednášek máte s řešením těchto úloh problém, tak se na nejasnosti co nejdříve ptejte na on-line konzultaci k přednášce, případně zvažte se mnou domluvit na konzultaci (zvlášť pokud tento stav u vás přetrvává i po dalších přednáškách).

Upozornění: Úlohy jsou vybrány a postaveny tak, abyste si po přednášce a před přednáškou následující mohli rychle připomenout hlavní části probrané látky a ověřit si její pochopení. Nicméně úlohy nejsou vyčerpávajícím přehledem látky z přednášek a tady rozhodně nepokrývají kompletní látku přednášek, která bude vyžadována u zkoušky. Pokud tedy u každé úlohy víte, jak by se měla řešit, tak to ještě neznamená, že jste dostatečně připraveni na zkoušku – nicméně jste jistě na velmi dobré cestě. Každopádně nezapomeňte, že na zkoušce se vyžaduje pochopení a porozumění právě všem konceptům ze všech přednášek, a navíc jsou zkouškové příklady postaveny komplexněji tak, aby ověřily také vaši schopnost přemýšlet nad látkou napříč jednotlivými přednáškami.