1) Jaký je rozdíl mezi paměťmi typu EPROM a EEPROM?

2) Jaký je rozdíl mezi paměťmi typu EEPROM a „flash“? Pro jaký druh přístupu je vhodnější použití paměti EEPROM a pro jaké paměti „flash“?

3) Srovnejte paměti EEPROM a „flash“ stran rychlosti s pamětmi SRAM a DRAM.

4) Předpokládejte, že by nám rychlost paměti EEPROM dostačovala pro použití jako operační paměti nějakého počítače. Bylo by vhodné ji použít místo paměti DRAM? Mělo by použití EEPROM oproti DRAM nějaké nevýhody?

5) Co znamená označení NVRAM? Jaké paměti byste do takové kategorie zařadili?

6) Jaká zařízení (řadiče) bude v sobě mít (může mít) zabudovaný typický jednočipový počítač?

7) Co je tzv. ADC převodník?

8) Co je tzv. DAC převodník?

9) Co je tzv. GPIO řadič? Připomeňte si, že s GPIO řadičem jsme již pracovali v 7. sadě self-assessment úloh, kde jsme jím ovládali 7-segmentový displej (kde jsme si koncept GPIO řadiče trochu vyspoilovali 😉).

10) Co znamenají termíny *sektor*, *stopa*, *cylinder*, *hlava* (*hlavička*) u pevného disku?

11) Jaká je typická velikost sektoru na pevném disku?

12) Jaké jsou 2 typické způsoby adresování sektorů na pevném disku?

13) Srovnejte pevné disky stran rychlosti s pamětmi DRAM.

14) Jaké druhy přístupů k posloupnosti sektorů na pevném disku jsou vhodné a jaké naopak nevhodné?

15) Jakém způsobem se standardně adresují sektory na CD nebo DVD disku?

16) Navrhněte, jak by mohlo vypadat komunikační rozhraní nějakého řadiče pevných disků (tzv. Hard Drive Controller, nebo-li **HDC**). Chceme, aby řadič podporoval práci s pevnými disky do velikosti 1 TB. Napište seznam všech registrů, které bychom u takového řadiče potřebovali – pro každý registr napište jeho velikost, napište, zda je R/O, W/O nebo R/W, a popište jeho funkci.

Dále předpokládejte že chceme v pseudokódu popsat funkci použití nějakého řadiče. Pro pseudokód využijeme základní syntaxi Pythonu, ale s rozšířením o operace čtení a zápisů z/na konkrétní adresu v operační paměti jako „*přečtení N bytů z adresy A*“, a o operace přímého čtení a zápisů z/do registrů nějakého řadiče připojeného k procesoru: „*zápis M bytů do registru R1 řadiče C*“ nebo „*přečtení M bytů z registru R2 řadiče C*“. Pokud bychom tedy takto chtěli pro RS-232 řadič, který známe z úvodu semestru, zapsat algoritmus „přečtení 1 bytu přijatého po RS-232 lince a jeho zapsání na adresu A“, tak bychom v našem pseudokódu mohl vypadat např. následujícím způsobem:

```
stav = 0
while stav == 0:
   stav = „čtení 1 bytu ze Status Registru řadiče RS-232“
data = „čtení 1 bytu z Data Registru řadiče RS-232“
„zápis 1 byte z data na adresu A“
```

a) Napište v uvedeném pseudokódu algoritmus „zápisu dat 1 sektoru (jehož data máme uložená v operační paměti na adrese X) na pevný disk, který je připojený k navrženému **HDC**. Data chceme zapsat do sektoru s číslem T“.
b) Napište v uvedeném pseudokódu algoritmus „přečtení 1 sektoru s číslem T z pevného disku připojeného k navrženému **HDC**, a uložení načtených dat do operační paměti od adresy X“.

17) Co je tzv. SSD disk? Bude pro typický SSD nějaký rozdíl v komunikaci s ním a v komunikaci s pevným diskem?

18) Vysvětlete termíny *bázová adresa* (*base address*) a *offset*.

19) Předpokládejte, že provedeme v Pythonu příklaz „`data = getData()`“, kde `getData` je nějaká funkce, která nám vrátí 5 bytů s hodnotami `0xAB` `0xCD` `0xFF` `0x00` `0x42` jako objekt typu `bytes`. Popište, co bude přesně

obsahovat proměnná `data`, a jak přesně bude vypadat vrácený objekt `bytes` v paměti (včetně všech data potřebných pro správu objektů v Pythonu). Předpokládejte 32-bytovou verzi Pythonu běžící na 32-bitové platformě, a že vrácený objekt typu `bytes` leží na adrese `0x0012A500`.

20) Co je soubor? Jaká hlavní metadata si o něm souborový systém musí pamatovat?

21) Co je souborový systém?

22) Předpokládejte soubory níže uvedených velikostí (z pohledu programu – pokud bychom např. přečetli celý jeho obsah, tak bychom získali právě takový počet bytů) – pro každou napište kolik bytů bude doopravdy soubor na disku zabírat, a vás výsledek zdůvodněte:

a) 189 B
b) 4096 B
c) 1 MB
d) 5,6 kB
e) 0 B (prázdný soubor) – zkuste vymyslet, jak by takový soubor byl asi v souborovém systému reprezentovaný

23) Jak se pozná, který sektor je na disku volný a mohou se do něj uložit data nějakého souboru?

24) Proč musíme v Pythonu soubor nejprve otevřít voláním `open`, a pak volat funkce `read` a `write` na vráceném objektu? Proč nemůžeme jméno souboru rovnou předávat každému volání `read`, resp. `write`?

25) Jaký je v Pythonu rozdíl mezi režimy otevření souboru `"r"` (případně `"w"`) a `"rb"` (případně `"wb"`)?

26) Můžeme textový soubor otevřít v režimu `"rb"`?

27) Co popisuje formát souboru? Co je obsahem popisu takového formátu?

28) Co je hlavička (header) souboru? Musí vždy začínat na offsetu 0?

29) Co je *signature* souboru, resp. jeho *magic konstanta*?

30) Připomeňte si formát obrazového souboru typu BMP – viz 11. přednáška, resp. viz části specifikace:
https://en.wikipedia.org/wiki/BMP_file_format#Bitmap_file_header
https://en.wikipedia.org/wiki/BMP_file_format#DIB_header_(bitmap_information_header), pouze část Windows BITMAPINFOHEADER
https://en.wikipedia.org/wiki/BMP_file_format#/media/File:BMPfileFormat.png

Prostudujte obsah souboru `11-Image1.bmp` (viz příloha těchto SA úloh) v nějakém hexvieweru nebo si zobrazte jeho hexdump, a zjistěte rozměry uloženého obrázku – tj. šířku (položka Bitmap/Image width in pixels) a výšku (položka Bitmap/Image height in pixels).

31) Prostudujte si specifikaci hlavičky obrazového souboru ve formátu PCX – viz:
https://en.wikipedia.org/wiki/PCX

Prostudujte obsah souboru `11-Image2-256colors-RLE.pcx` (viz příloha těchto SA úloh) v nějakém hexvieweru nebo si zobrazte jeho hexdump, a zjistěte rozměry uloženého obrázku – tj. šířku (hodnota *maximum x co-ordinate* + 1) a výšku (hodnota *maximum y co-ordinate* + 1).

32) Napište v Pythonu program, který se uživatele zeptá na jméno souboru. Po zadání jména program přečte obsah tohoto souboru (alespoň jeho relevantní část), a pokusí se ho interpretovat jako soubor ve formátu PCX (viz otázka 31). Program poté z obsahu souboru zjistí šířku a výšku v souboru uloženého obrázku a tyto jako celá čísla v desítkové soustavě vypíše uživateli. Pokud se vám v programu podaří detekovat, že zadaný soubor rozhodně není ve formátu PCX, tak má program vypsat „`Not a PCX image.`“. Pro jednoduchost neřešte a neošetřujte jiné chybové stavy.

Poznámka: V dále uvedených úlohách předpokládejte, že používáme dosud na přednáškách uvažovanou konvenci, že textové řetězce jsou posloupnost znaků, kde každý 1 znak je identifikován právě jedním 1 byte kódem (bezznaménkovým 8-bitovým celým číslem). Předpokládejte, že **kód znaku „mezera“ je `0x20`**.

Poznámka: Ke zkoušce z Principů počítačů není třeba znát přesné fungování žádného konkrétního souborového systému – nicméně byste měli rozumět obecným principům, jak bude souborový systém navržen, tak jak to známe

z přednášek. Z toho vyplývá, že byste měli být schopni i pochopit popis nějakého vám dosud neznámého souborového systému a na základě takového popisu s ním pracovat.

Souborový systém FAT: Operační systém Windows mimo jiné podporuje práci se souborovým systémem FAT (který se původně rozšířil s operačním systémem MS-DOS, a který nyní mimo jiné podporuje i většina dalších operačních systémů jako Linux, macOS, apod.) – jelikož detaily návrhu tohoto souborového systému neznáme z přednášky (a není třeba je ani znát ke zkoušce – viz poznámka výše), tak si ho stručně představme (ve zjednodušené podobě) – všechny hodnoty jsou v systému FAT uložené **v LE pořadí**. Disk naformátovaný souborovým systémem FAT je rozdělen do následujících skupin sektorů (číslováno od 0 v LBA adresování):

a) Sektor 0 obsahuje základní informace o souborovém systému, tzv. *BIOS Parameter Block* (BPB) – jeho obsah ale pro jednoduchost v níže uvedených úlohách a vysvětlení používat nebudeme.
b) Sektory 1 až X-1 obsahují další metadata souborového systému, která již přímo popisují informace o obsazenosti jednotlivých „datových sektorů“, a metadata o jednotlivých v souborovém systému uložených souborech – více viz dále. Hodnotu X lze spočítat z informací uložených v BIOS Parameter Block, nicméně v úlohách dále máte hodnotu X rovnou uvedenu.
c) Sektory X až N-1 (kde N je celkový počet všech sektorů disku) jsou datové sektory, a jsou buď volné nebo obsahují data nějakého souboru. Pozor: souborový systém FAT má několik zvláštností:
    1) Souborům jsou obecně přidělovány skupiny datových sektorů, kterým FAT říká **cluster**y. Každý cluster obsahuje vždy stejný počet datových sektorů – níže uvedené **úlohy používají** souborový systém FAT zformátovaný s nastavením **1 cluster = 1 datový sektor**, proto dále můžete považovat cluster a datový sektor za synonyma.
    2) Clustery (datové sektory) jsou číslovány celými čísly od 2 – tj. „první cluster“ má číslo 2, a v případě konvence 1 cluster = 1 datový sektor je právě sektorem číslo X (v LBA adresování od začátku disku).

Struktura metadat souborového systému FAT:

Sektory 1 až F-1 disku (kde F < X) obsahují tzv. **tabulku FAT**- ta pro každý cluster (datový sektor) obsahuje 1 záznam = 1 bezznaménkové číslo, které popisuje informace o obsazenosti daného clusteru. Pokud má tento záznam hodnotu 0, tak je daný cluster volný. Dále budeme uvažovat pouze variantu **FAT16**, která pro uvedené záznamy používá **16-bitová čísla**. **Pozor:** přestože v souborovém systému neexistují clustery s číslem 0 a 1 (viz výše číslování clusterů od 2), tak tabulka FAT začíná dvěma záznamy právě pro takový fiktivní cluster 0 a 1 – tedy až třetí záznam FAT je doopravdy užitečný a obsahuje informaci o clusteru číslo 2 (prvním skutečně existujícím clusteru na disku).

Sektory R až X-1 disku (kde R >= F) obsahují tzv. **kořenový adresář** (**root directory**), který obsahuje posloupnost 32-bytových záznamů – viz https://en.wikipedia.org/wiki/Design_of_the_FAT_file_system#Directory_entry – stručně: jeden takový 32-bytový záznam vždy popisuje informace o 1 souboru uloženém v souborovém systému – mimo jiné:

a) Jméno souboru (1 až 8 znaků – nepoužitá místa jména jsou do 8 vyplněna znakem mezera), příponu jména souboru (0 až 3 znaky – nepoužitá místa jsou do 3 vyplněna znakem mezera) – jako platné znaky mohou být velká písmena anglické abecedy, číslice, znaky `! # $ % & ' ( ) - @ ^ _ ` { } ~`.
b) Velikost souboru v bytech (32-bitové bezznaménkové číslo).
c) Číslo (16-bitové bezznaménkové) prvního clusteru, který obsahuje data daného souboru.

Jak zjistit seznam všech clusterů, ve kterých jsou uložena data nějakého souboru?

Pro každý soubor si souborový systém FAT16 pamatuje seznam všech jeho datových clusterů ve formě jednosměrně vázaného seznamu, a to následujícím způsobem:

0) Číslo C1 prvního clusteru souboru najdeme v záznamu kořenového adresáře pro tento soubor.
1) Podíváme do FAT (File Allocation Table) tabulky na záznam pro cluster číslo C1, a zde najdeme buď:
    a) Hodnotu z rozsahu `0xFFF8` až `0xFFFF`, pak je cluster číslo C1 prvním, ale současně posledním clusterem souboru.
    b) Nenulovou hodnotu C2 menší než `0xFFF8`, pak je cluster číslo C2 druhým clusterem souboru.
2) Podíváme do FAT tabulky na záznam pro cluster číslo C2, a zde najdeme buď:
    a) Hodnotu z rozsahu `0xFFF8` až `0xFFFF`, pak je cluster číslo C2 posledním clusterem souboru – tedy soubor má data uložena právě v clusterech C1 a C2.
    b) Nenulovou hodnotu C3 menší než `0xFFF8`, pak je cluster číslo C3 třetím clusterem souboru.
3) Podíváme do FAT tabulky na záznam pro cluster číslo C3, a zde najdeme buď:

   a) Hodnotu z rozsahu `0xFFF8` až `0xFFFF`, pak je cluster číslo C3 posledním clusterem souboru – tedy soubor má data uložena v clusterech C1, C2, C3.
   b) Nenulovou hodnotu C4 menší než `0xFFF8`, pak je cluster číslo C4 čtvrtým clusterem souboru.
4) Pokračujeme dále analogicky krokům 1 až 3 tak dlouho, dokud tímto postupem nezískáme celý seznam čísel všech clusterů C1, C2, C3, C4, …, Cn, které obsahují data souboru.

Poznámka: Soubory v souborovém systému FAT mohou být fragmentované, tj. posloupnost čísel C1 až Cn nemusí být souvislá, a navíc **nemusí platit**, že $C_{i+1} > C_i$.

Detailnější popis samotné FAT tabulky naleznete na (sledujte informace relevantní pro variantu FAT16):
https://en.wikipedia.org/wiki/Design_of_the_FAT_file_system#Cluster_map

Společná část k otázkám 33 až 36: V příloze najdete soubor `11-Fat16DiskImageVer1.bin`, který obsahuje obraz disku (disk image – viz doplnění 11. přednášky na 11. konzultaci v čase 1:02:48 až 1:17:11) naformátovaného souborovým systémem FAT16 s následujícími parametry:

a) velikost sektoru je 512 bytů, a 1 cluster = 1 datový sektor,
b) velikost disku je 4 MiB – z toho vyplývá hodnota **N** (viz část Souborový systém FAT výše),
c) hodnota **F** (určující velikost FAT tabulky) je **33** (zajímavost: od sektoru 33 obsahuje souborový systém redundantní zcela identickou kopii tabulky FAT),
d) hodnota **R** (sektor, kde začíná kořenový adresář) je **65**,
e) hodnota **X** (sektor, který je prvním datovým sektorem, tj. clusterem číslo 2) je **97**.

Poznámka: V příloze najdete též soubor `11-Fat16DiskImageVer1-JustRootDirectory.bin`, který pro přehlednost obsahuje pouze kopii prvního sektoru kořenového adresáře z výše uvedeného obrazu disku:

```
00000000: 41 42 43 20 20 20 20 20|54 58 54 20 00 36 66 79  | ABC     TXT .6fy
00000010: 95 51 95 51 00 00 66 79|95 51 03 00 28 00 00 00  | .Q.Q..fy.Q..(...
00000020: 54 49 54 4C 45 20 20 20|54 58 54 20 00 94 19 7A  | TITLE   TXT ...z
00000030: 95 51 95 51 00 00 19 7A|95 51 21 00 28 00 00 00  | .Q.Q...z.Q!.(...
00000040: 42 49 4F 20 20 20 20 20|4D 44 20 20 00 6F 71 79  | BIO     MD  .oqy
00000050: 95 51 95 51 00 00 71 79|95 51 05 00 46 03 00 00  | .Q.Q..qy.Q..F...
00000060: 42 4F 4F 4B 20 20 20 20|54 58 54 20 00 5F AB 79  | BOOK    TXT ._Ťy
00000070: 95 51 95 51 00 00 AB 79|95 51 07 00 16 0C 00 00  | .Q.Q..Ťy.Q...↑..
00000080: 48 41 52 52 59 20 20 20|54 58 54 20 00 88 7A 79  | HARRY   TXT ..zy
00000090: 95 51 95 51 00 00 7A 79|95 51 09 00 24 00 00 00  | .Q.Q..zy.Q..$...
000000A0: 44 41 54 41 20 20 20 20|54 58 54 20 00 2F 80 79  | DATA    TXT ./.y
000000B0: 95 51 95 51 00 00 80 79|95 51 00 00 00 00 00 00  | .Q.Q...y.Q......
000000C0: 4C 4F 4E 47 49 4E 46 4F|54 58 54 20 00 C5 84 79  | LONGINFOTXT .Ĺ.y
000000D0: 95 51 95 51 00 00 84 79|95 51 0A 00 34 08 00 00  | .Q.Q...y.Q..4...
000000E0: 54 48 49 4E 47 53 20 20|20 20 20 10 00 40 DC 79  | THINGS     ..@Üy
000000F0: 95 51 95 51 00 00 DC 79|95 51 12 00 00 00 00 00  | .Q.Q..Üy.Q......
00000100: 45 4D 50 54 59 20 20 20|20 20 20 10 00 02 05 7A  | EMPTY      ....z
00000110: 95 51 95 51 00 00 05 7A|95 51 1B 00 00 00 00 00  | .Q.Q...z.Q......
00000120: E5 4F 54 54 45 52 20 20|54 58 54 20 00 71 0C 7A  | ÍOTTER  TXT .q↑z
00000130: 95 51 95 51 00 00 0C 7A|95 51 1C 00 81 09 00 00  | .Q.Q..↑z.Q......
00000140: 55 53 45 52 53 20 20 20|20 20 20 10 00 7E 2F 7A  | USERS      ..~/z
00000150: 95 51 95 51 00 00 2F 7A|95 51 22 00 00 00 00 00  | .Q.Q../z.Q"".....
00000160: 48 41 47 52 49 44 20 20|4D 44 20 20 00 40 24 7A  | HAGRID  MD  .@$z
00000170: 95 51 95 51 00 00 24 7A|95 51 23 00 EC 02 00 00  | .Q.Q..$z.Q#.ě...
00000180: 53 45 43 52 45 54 20 20|20 20 20 20 00 67 2B 7A  | SECRET      .g+z
00000190: 95 51 95 51 00 00 2B 7A|95 51 25 00 04 00 00 00  | .Q.Q..+z.Q%.....
000001A0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  | ................
000001B0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  | ................
000001C0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  | ................
000001D0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  | ................
000001E0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  | ................
000001F0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  | ................
```

Zde si můžeme všimnout, že např. data souboru `BIO.MD` začínají v clusteru (datovém sektoru) číslo 5.

Poznámka: V příloze též najdete soubor `11-Fat16DiskImageVer1-JustFat1Sector.bin`, který pro přehlednost obsahuje pouze kopii prvního sektoru FAT tabulky z výše uvedeného obrazu disku (v hexdumpu níže je vynechaná interpretace bytů jako 1 bytových znaků, jelikož ta je pro pochopení FAT tabulky irelevantní; navíc není zobrazený celý obsah souboru, ale pouze důležitá část obsahující nějaké nenulové byty):

```
00000000: F8 FF FF FF 00 00 FF FF|00 00 06 00 FF FF 08 00
00000010: 0F 00 FF FF 0B 00 0C 00|0D 00 0E 00 FF FF 10 00
00000020: 11 00 13 00 FF FF 14 00|FF FF FF FF FF FF FF FF
00000030: FF FF FF FF FF FF FF FF|00 00 00 00 00 00 00 00
00000040: 00 00 FF FF FF FF 24 00|FF FF FF FF FF FF 00 00
00000050: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00
```

V **červeně orámečkované části** si můžeme všimnout, že pro fiktivní clustery 0 a 1 obsahuje FAT tabulka hodnoty `0xFFF8` a `0xFFFF` (tedy se tváří jako konec nějakého souboru – nicméně víme, že clustery 0 a 1 na disku doopravdy neexistují – viz popis FAT výše – tedy jsou pro nás tyto hodnoty irelevantní). Dále si můžeme všimnout, že cluster 2 je volný (jeho záznam ve FAT obsahuje hodnotu `0x0000`).

Jelikož jsme z obsahu kořenového adresáře zjistili, že soubor `BIO.MD` začíná v clusteru 5, tak ze **zeleně orámečkované části** můžeme zjistit, že celkově jsou data souboru `BIO.MD` uložena právě v clusterech 5 a 6 (jelikož záznam ve FAT pro cluster 5 obsahuje hodnotu `0x0006`, a záznam pro cluster 6 hodnotou `0xFFFF` značí, že se jedná o poslední cluster souboru).

Poznámka: Při řešení úloh níže použijte nějaký vhodný hexviewer/hexeditor, který je k dispozici pro váš operační systém. Na 11. konzultaci jsme si např. ukazovali webový hexeditor: https://hexed.it/
Nebo na *Windows* můžete použít např.:

- http://www.pspad.com/ – obecný editor souborů pro Windows, který umožňuje i v hex módu zobrazení a editaci libovolných souborů
- https://mh-nexus.de/en/hxd/ – **pozor:** velmi „mocný nástroj“, který vám umožňuje editovat i data přímo na vašem disku (tj. můžete si omylem poškodit metadata vašeho souborového systému, a přijít o přístup k vašim souborům, nebo znemožnit spuštění/boot vašeho operačního systému)

Na *macOS* např.:

- https://ridiculousfish.com/hexfiend/

Na *Linuxu* např.:

- Nástroj `xxd` na příkazové řádce
- https://wiki.gnome.org/Apps/Ghex – hexeditor pro GNOME

33) Napište seznam čísel clusterů, ve kterých jsou uložena data souboru `BOOK.TXT`.

34) Jaké slovo je tajným heslem (`SECRET`)?

35) Jaké anglické slovo je v souboru `BOOK.TXT` uloženo od offsetu `1019`?

36) Napište v Pythonu program, který přečte obsah souboru `11-Fat16DiskImageVer1-JustRootDirectory.bin` obsahujícího data z 1 sektoru kořenového adresáře v souborovém systému FAT16, a pro všechny běžné soubory (tj. položky kořenového adresáře, které nemají nastavený ani jeden z příznaků *Subdirectory*, *Volume Label*, *Device* v položce *File Attributes*), které nejsou smazané (= první byte jména souboru není ani `0x00`, ani `0xE5`), vypíše jejich velikost v bytech jako celé číslo v desítkové soustavě (pro každý soubor na zvláštním řádku). Pro jednoduchost neřešte a neošetřujte chybové stavy.

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.