Bakalářské zkoušky (příklady otázek z informatiky)

podzim 2022

1 Průnik regulárních výrazů (3 body)

1. Uveďte definici regulárního výrazu. Zformulujte Kleeneho větu.
2. Mějme následující regulární výrazy nad abecedou $\Sigma = \{a, b\}$:

$$R_1 = ((a + b)(a + b)(a + b))^*$$
$$R_2 = (a + b)^* a$$

Zkonstruujte *deterministický* konečný automat A přijímající průnik jazyků popsaných výrazy R_1 a R_2, tj. takový, že platí:

$$L(A) = L(R_1) \cap L(R_2)$$

2 Silná souvislost (3 body)

1. Definujte, co je silně souvislá komponenta orientovaného grafu $G = (V, E)$.
2. Popište co nejrychlejší algoritmus, který spočítá rozklad G na jeho silně souvislé komponenty.
3. Policie v Kocourkově udělala ze všech ulic jednosměrky. Starosta nejprve tvrdil, že se přesto dá (legálně) z každé křižovatky dostat na každou jinou křižovatku, ale po předložení protipříkladu své tvrzení zeslabil. Řekněme, že křižovatka x je *dobrá*, pokud z každé křižovatky y, která je z x dosažitelná, se dá dojet zpět do x. Starosta tvrdí, že 95% křižovatek v Kocourkově je dobrých. Navrhněte algoritmus, který jeho tvrzení potvrdí nebo vyvrátí. Ideálně by měl algoritmus běžet v lineárním čase.

3 SQL dotazování (3 body)

Uvažujte následující relační schéma:

- Herec (IDU, jmeno, prijmeni, narodnost, vek), IDU je klíč
- Film (ID, nazev, zanr, barva), ID je klíč
- Hraje (IDU, ID), IDU a ID je klíč, IDU je podmnožinou Herec.IDU, ID je podmnožinou Film.ID

V SQL vyjádřete příkazy (dotazy):

1. Přidejte záznamy o filmu 'The Imitation Game' a jeho herci jménem Benedict Cumberbatch. (Zbylá data vymyslete.)
2. Vytvořte tabulku režisér (s atributy ID, jméno, příjmení) a doplňte příslušný vztah k filmům (tj. které filmy režisér režíruje). Zajistěte referenční integritu.
3. Vypište jména herců, kteří hrají ve velkofilmech (filmech, ve kterých hraje více než 200 herců).
4. Odstraňte sloupec barva u filmů.

4 Programování - grafy (3 body)

Pro tuto otázku si vyberte libovolný mainstreamový objektově orientovaný staticky typovaný jazyk (C++, C# nebo Java) a svůj výběr napište na začátek řešení.

1. Naimplementujte/nadeklarujte vhodné objektové rozhraní (pomocí interfaces nebo abstraktních tříd) pro neorientované grafy $G = (V, E)$ ve své nejběžnější podobě (hrany spojují právě dva vrcholy, mezi dvěma vrcholy vede nejvýš jedna hrana). Motivací je, aby toto rozhraní mohly implementovat různé reprezentace grafů (seznam sousedů, matice sousednosti, ...), zatímco programátor grafových algoritmů (BFS, hledání min. kostry, ...) může napsat svůj kód pouze jednou a bude fungovat se všemi reprezentacemi grafů.

   Vámi definované rozhraní by mělo pokrývat 3 entity (*vrchol*, *hranu* a *graf*), přičemž *vrchol* nabídne přístup ke všem incidentním hranám, *hrana* nabídne přístup ke svým dvěma vrcholům a *graf* umožní přístup k seznamu všech vrcholů a seznamu všech hran. Modifikace grafu v tomto rozhraní neuvažujeme (struktura grafu je pouze pro čtení).

   Každý vrchol a každá hrana má příznak (pro jednoduchost mu říkejme *tag*). Tag u vrcholu může například indikovat, zda byl vrchol již navštíven, tag u hrany může například značit její váhu. Tagy tedy mohou mít obecně různé datové typy, avšak pro danou instanci grafu mají všechny vrcholy stejný typ tagu a všechny hrany taktéž. Váš interface by měl být obecný, aby si programátor mohl při jeho implementaci zvolit typy tagů pro vrcholy a pro hrany. Interface také nabídne pro každý vrchol a hranu vhodné prostředky, jak tag přečíst a nastavit.

2. Představme si implementaci vašeho rozhraní, kde tagy vrcholů značí index komponenty souvislosti (typu `int`, indexované od 0) a tagy hran jsou typu `float` (případně podobného typu ve vašem zvoleném jazyce) a značí jejich délky. Tagy hran jsou na počátku již inicializované, tagy vrcholů musíte nastavit vy.

   Napište kód těla funkce využívající vaše rozhraní, která na vstupu dostane graf a reálné číslo r. Váš kód provede rozklad grafu na komponenty souvislosti tak, že každý vrchol bude mít svůj index komponenty uložený jako tag (konkrétní hodnoty indexů nejsou podstatné, důležité je, že vrcholy ve stejné komponentě mají stejný tag). Hrany, které mají větší délku (tag) než stanovený parametr r, nebereme při určování komponent v potaz (tedy jako by vůbec neexistovaly).

5 Virtuální paměť (3 body)

Uvažujte architekturu procesoru s podporou stránkování a délkou virtuální a fyzické adresy 32 bitů. Pro jednoduchost předpokládejte, že k překladu adres procesor používá jednoúrovňovou stránkovací tabulku. Velikost stránek je 4 KiB. Vedle absolutně nezbytných atributů je možné pro jednotlivé stránky nastavit, zda je smí proces modifikovat a zda je možné v nich spouštět kód. Procesor také pro jednotlivé stránky poskytuje informaci o tom, zda se k dané stránce přistupovalo a zda do ní bylo zapsáno.

Část A

1. Navrhněte a schematicky znázorněte formát položky stránkovací tabulky a vysvětlete význam jednotlivých polí. Položka stránkovací tabulky musí obsahovat všechny nutné informace a musí být pro procesor efektivně přístupná (maximálně 1 čtení z paměti).
2. Kolik položek bude mít stránkovací tabulka a kolik paměti bude tato tabulka zabírat pro každý spuštěný proces?

Část B

1. Napište (v pseudokódu) funkci `create_pte`, která pro zadanou fyzickou adresu a atributy stránky vrátí položku stránkovací tabulky. Funkce by měla mít zhruba následující signaturu:

   ```
   pte_t create_pte(address_t phys_addr, bool present, bool writable, bool executable)
   ```

   Typ `address_t` je celočíselný typ, který postačuje k uložení fyzické adresy předané v parametru `phys_addr`. Typ `pte_t` je celočíselný typ, který reprezentuje Vámi navrženou položku stránkovací tabulky (a splňuje kritérium pro efektivní přístup).

2. Napište (v pseudokódu) funkci `set_mapping`, která do stránkovací tabulky pro zadanou virtuální adresu vloží požadovaný záznam. Funkce by měla mít zhruba následující signaturu:

   ```
   void set_mapping(pte_t[] page_table, address_t virt_addr, pte_t entry)
   ```

   Předpokládejte, že stránkovací tabulka je reprezentována jako pole položek typu `pte_t`, takže s parametrem `page_table` můžete zacházet jako s polem. Parametr `virt_addr` představuje virtuální adresu, pro kterou máte do tabulky vložit položku v parametru `entry`.

Část C

Předpokládejte, že **souvislý** blok 4 stránek virtuální paměti od adresy `0x0000B000` byl namapován na **souvislý** blok fyzické paměti. Blok reprezentuje část heapu procesu, obsah paměti je tedy možné číst a modifikovat, ale není možné v něm

spouštět kód. Předpokládejte také, že proces **přečetl** obsah proměnné (o velikosti 4 B) z virtuální adresy `0x0000CE90`, což bylo přeloženo na přístup do fyzické paměti na adrese `0xA1018E90`. Pozměněnou hodnotu proměnné pak proces **zapsal** na virtuální adresu `0x0000D854`.

1. Uveďte, na kterých fyzických adresách budou uloženy položky stránkovací tabulky pro výše uvedený blok paměti, pokud víte, že stránkovací tabulka procesu je uložena v souvislém bloku fyzické paměti od adresy `0x13400000`.
2. Uveďte (hexadecimální) hodnoty všech položek stránkovací tabulky pro výše uvedený rozsah virtuální paměti, včetně všech atributů.

6 Směrovací tabulka (3 body)

Mějme standardní směrovač (router) používající následující směrovací tabulku:

Destination	Netmask	Interface	Gateway	Metric
195.113.19.0	255.255.255.0	(A) 195.113.19.20	on-link	1
172.217.23.0	255.255.255.0	(B) 172.217.23.55	on-link	1
216.58.0.0	255.255.0.0	(C) 216.58.201.78	on-link	1
185.17.100.0	255.255.255.0	(B) 172.217.23.55	(D) 172.217.23.111	10
185.17.200.0	255.255.255.0	(B) 172.217.23.55	(E) 172.217.23.222	10
104.0.0.0	255.0.0.0	(C) 216.58.201.78	(F) 216.58.90.100	10

Nejprve vysvětlete význam jednotlivých sloupců ve směrovací tabulce.

Následně na síťovém rozhraní (A) 195.113.19.20 postupně přijmeme následující standardní IPv4 datagramy:

1. Datagram pro příjemce (G) 185.17.200.50
2. Datagram pro příjemce (A) 195.113.19.20
3. Datagram pro příjemce (F) 216.58.90.100
4. Datagram pro příjemce (H) 74.6.231.21
5. Datagram pro příjemce (I) 104.103.85.139
6. Datagram pro příjemce (J) 255.255.255.255
7. Datagram pro příjemce (K) 216.58.255.255

Pro zjednodušení naší práce budeme všechny použité IP adresy symbolicky označovat pomocí písmen.

Pro každou z předchozích situací určete, jakým způsobem bude daný datagram zpracován. Pokud bude posílán dále, jakým směrem a na jakou MAC (EUI-48) adresu v rámci linkové L2 vrstvy? Myslí se tím konkrétní dobře známá MAC adresa, v opačném případě MAC adresa odpovídající našim IP adresám (A) až (K). Vysvětlete a odůvodněte.

7 Aproximační algoritmy (AO) (3 body)

1. Popište 8/7-aproximační pravděpodobnostní algoritmus pro 3SAT, a dokažte uvedený odhad na aproximační faktor.
2. Popiště aproximační algoritmus pro SAT založený na lineárním programování a uveďte jeho aproximační faktor.
3. Pokud znáte ještě nějaký lepší algoritmus než ten z bodu 2, popište ho.

8 Voroného diagramy (AO) (3 body)

Nechť P je konečná množina bodů v obecné poloze v $\mathbb{R}^d$ a $\mathrm{conv}(P)$ značí její konvexní obal.

1. Definujte region $\mathrm{reg}(p)$ bodu $p \in P$ a Voroného diagram množiny P.
2. Navrhněte (libovolný) algoritmus, který pozná pro $x \in \mathbb{R}^d$ a $p \in P$, zda x patří do $\mathrm{reg}(p)$ a co nejlépe asymptoticky odhadněte časovou složitost navrženého algoritmu. (Můžete předpokládat, že aritmetické operace nad reálnými čísly jsou prováděny v jednotkovém čase.)

Dále rozhodněte a zdůvodněte, zda platí:

3. Průnik dvou regionů je vždy konvexní mnohostěn.

4. Průnik dvou regionů je vždy konvexní polyedr.

9 Kódy (AO) (3 body)

1. Zadefinujte následující pojmy: Kód nad abecedou Σ, délka, velikost a minimální vzdálenost kódu.
2. Nechť C je kód nad abecedou Σ délky ℓ, velikosti s a minimální vzdálenosti $d \geq 2$. Nechť C' je kód vzniklý z C tak, že z každého kódového slova smažeme poslední znak. Jaká je délka a velikost C'? Ukažte, že minimální vzdálenost C' je alespoň $d - 1$.
3. Ukažte, že kód délky ℓ nad abecedou Σ a s minimální vzdáleností d má velikost nejvýše $|\Sigma|^{\ell-d+1}$.

10 Kódy (DMS) (3 body)

1. Zadefinujte následující pojmy: Kód nad abecedou Σ, délka, velikost a minimální vzdálenost kódu.
2. Nechť C je kód nad abecedou Σ délky ℓ, velikosti s a minimální vzdálenosti $d \geq 2$. Nechť C' je kód vzniklý z C tak, že z každého kódového slova smažeme poslední znak. Jaká je délka a velikost C'? Ukažte, že minimální vzdálenost C' je alespoň $d - 1$.
3. Ukažte, že kód délky ℓ nad abecedou Σ a s minimální vzdáleností d má velikost nejvýše $|\Sigma|^{\ell-d+1}$.

11 Ramseyova teorie (DMS) (3 body)

1. Zformulujte Ramseyovu větu pro grafy, s libovolným počtem barev.
2. Nechť $R(a, b)$ je nejmenší přirozené číslo takové, že každý graf s $R(a, b)$ vrcholy obsahuje kliku velikosti a nebo nezávislou množinu velikosti b. Ukažte, že $R(a, b) \leq R(a - 1, b) + R(a, b - 1)$.
3. Ukažte, že pro každé přirozené číslo $b \geq 1$ existuje graf bez trojúhelníků barevnosti alespoň $(R(3, b + 1) - 1)/b$.

12 Párování v grafech (DMS) (3 body)

1. Zformulujte Tutteovu větu o existenci perfektního párování.
2. Která z následujících tvrzení platí a proč?
   (a) Každý hranově 2-souvislý 3-regulární graf má perfektní párování.
   (b) Každý hranově 4-souvislý 4-regulární graf má perfektní párování.
   (c) Má-li graf G s n vrcholy perfektní párování, pak každá nezávislá množina v G má velikost nejvýše $n/2$.
   (d) Má-li největší nezávislá množina v grafu G s n vrcholy velikost nejvýše $n/2$, pak G má perfektní párování.

   Odpovědi zdůvodněte.

13 Základní formalismy pro popis přirozeného jazyka (ML) (3 body)

1. Vyjmenujte a popište tři základní komponenty transformační gramatiky
2. Vysvětlete pojem transformace v Chomského transformační gramatice – kterými dvěma složkami je transformace definována?
3. Proč je nutné zavádět typy sestav rysů v unifikačních gramatikách?

14 Morfologická, syntaktická a sémantická analýza přirozeného jazyka (ML) (3 body)

1. Co je morfologická značka? Uveďte alespoň pět rysů, které bývají často zakódovány v množinách morfologických značek.
2. Popište závislostní strom, složkový strom a rozdíly mezi nimi.
3. Nakreslete (a) složkový strom a (b) závislostní strom pro větu *Marie píše dopis Janovi.*

15 Jazykové modelování (ML) (3 body)

1. Uvažujte následující jednoduchý korpus: *the cat cut the hat* Kolik různých znakových bigramů se v korpusu vyskytuje? Mezera je také znak. Jaká je jejich četnost v korpusu?
2. Popište hlavní myšlenku N-gramového jazykového modelování v počítačovém zpracování přirozného jazyka. Napište vzorec pro 3-gramový jazykový model.
3. Proč potřebujeme v jazykovém modelování vyhlazování (*smoothing*)?

16 HDR grafika (PG) (otázka studijního zaměření – 3 body)

Otázka se týká HDR (High Dynamic Range) grafiky, principů a aplikací.

1. Z jakých důvodů a v jakých oblastech grafiky se používá HDR grafika? Uveďte příklady, kdy nám může pomoci nejvíce.
2. Jak se HDR obraz reprezentuje v paměti počítače a v jakém formátu ho můžeme zapsat na disk? Naznačte postup pořízení HDR obrázku s pomocí fotoaparátu a stativu.
3. Jak se HDR obraz prezentuje pozorovateli na běžném (LDR) výstupním zařízení? (není potřeba uvádět nějaké komplikované postupy, jen popsat princip)

17 Scanline vyplňování (PG) (otázka studijního zaměření – 3 body)

Budeme se zabývat vybarvením vnitřku rovinného mnohoúhelníka v rastrovém prostředí (čtvercová mřížka pixelů), obrys útvaru není potřeba obkreslovat.

1. Jak by měl být mnohoúhelník zadán a jaké možnosti definice jeho vnitřku mají smysl?
2. Popište základní princip algoritmu řádkového vyplňování 2D mnohoúhelníka (uveďte i případné pomocné datové struktury používané v průběhu vyplňování).
3. Jak by se algoritmus zjednodušil, kdyby měl vyplňovat pouze konvexní útvary?

18 3D scéna pro kreslení na GPU (PG) (otázka studijního zaměření – 3 body)

1. Jak byste v paměti počítače reprezentovali 3D scénu, která se pak bude v reálném čase vykreslovat na grafické kartě (GPU)? Nemusíte psát přímo deklarace dat, ale popište dostatečně podrobně svůj návrh slovy.
2. Uvažujte systém hierarchické reprezentace tak, aby se komponenty (objekty, modely) daly jednoduše ve scéně opakovat bez toho, aby se musela data duplikovat.
3. Jak technicky implementovat animaci (pohyb) jednotlivých objektů ve scéně? Omezte se na nedeformovatelná tělesa ("rigid body animation"), která se jen ve scéně přemisťují otáčením a translací.

19 Základní koncepty profilování výkonu (SP) (otázka studijního zaměření – 3 body)

Toto je (naivní) program pro výpočet čísla π částečným součtem řady:

```
1  #include <climits>
2  #include <iostream>
3
4  int main (void) {
5      int i = 1;
6      int s = 1;
7      double x = 0;
8      while (i < INT_MAX) {
9          x += s * ((double) 1)/i;
10         i += 2;
11         s *= -1;
12     }
13     std::cout << x * 4 << std::endl;
```

```
14      return (0);
15  }
```

Cyklus výpočtu přeložil překladač pro procesor řady x86 takto:

```
1           pxor        %xmm0, %xmm0
2           movl        $1, %edx
3           movl        $1, %eax
4   .L2:
5           pxor        %xmm1, %xmm1
6           cvtsi2sdl   %edx, %xmm1
7           pxor        %xmm2, %xmm2
8           cvtsi2sdl   %eax, %xmm2
9           divsd       %xmm2, %xmm1
10          addsd       %xmm1, %xmm0
11          addl        $2, %eax
12          negl        %edx
13          cmpl        $2147483647, %eax
14          jne         .L2
```

Sémantiku instrukcí odhadněte z názvu a funkce programu, zkratky `add`, `div`, `neg` a `xor` označují příslušné aritmetické a logické operace, `cmp` je porovnání, `mov` je přesun hodnoty, `cvtsi2sdl` je přesun s převodem integer na double.

1. Měření ukazují, že procesor vykonal uvedený kód s zhruba IPC 2.5. Vyznačte ve výpisu skupiny instrukcí, v rámci kterých se mohou všechny instrukce vykonat paralelně, tak, abyste mohli paralelním vykonáváním instrukcí vysvětlit alespoň IPC 2.
2. Uvedený kód lze snadno paralelizovat například tak, že iterace cyklu rozdělíme rovnoměrně mezi n vláken. Měření ukazují, že při použití 8 vláken (na 8 nezávislých procesorech) bylo zrychlení programu 7. Odhadněte celkový počet instrukcí (přes všechna vlákna), které bylo potřeba vykonat pro zajištění koordinace vláken. Váš postup vysvětlete.

20 Paralelismus a synchronizace (SP) (otázka studijního zaměření – 3 body)

1. V některém z jazyků C, C#, C++ nebo Java implementujte spin lock. Ze synchronizačních operací zvoleného jazyka může vaše implementace používat pouze standardní atomické operace (včetně běžných operací typu read-modify-write). Ty v implementaci vyznačte.
2. Ve stejném jazyce implementujte ještě rekurzivní zámek s pasivním čekáním. Můžete použit spin lock z předchozího bodu a běžné kolekce. Rozhraní pro pasivní čekání si sami navrhněte a popište jeho sémantiku.

21 Testování funkčnosti (SP) (otázka studijního zaměření – 3 body)

Uvažujte následující jednoduchou implementaci zásobníku:

```
1   public class Stack {
2
3       private int [] data;
4       private int top;
5
6       public Stack (int capacity) {
7           if (capacity < 0) {
8               throw new NegativeArraySizeException ("Creating␣negative␣size␣stack");
9           }
10          data = new int [capacity];
11          top = 0;
12      }
13
14      public void push (int item) {
15          if (top == data.length) {
16              throw new ArrayIndexOutOfBoundsException ("Push␣to␣a␣full␣stack");
17          }
```

```
18          data [top] = item;
19          top ++;
20      }
21
22      public int pop () {
23          if (top == 0) {
24              throw new ArrayIndexOutOfBoundsException ("Pop␣from␣an␣empty␣stack");
25          }
26          top --;
27          return (data [top]);
28      }
29  }
```

1. Napište definici pokrytí vhodnou pro jednotkové testování uvedené implementace. Vaše definice by měla zahrnovat vzorec pro výpočet procentuálního pokrytí a význam použitých symbolů.
2. Načrtněte, jaké jednotkové testy jsou potřeba pro úplné pokrytí uvedené implementace podle vámi zvolené definice. Jednotlivé testy popište pseudokódem nebo popisem toho, co konkrétně mají testovat.
3. Napište jednu další definici pokrytí a rozhodněte, která z Vámi uvedených definic pokrytí je pro uvedenou implementaci vhodnější. Své rozhodnutí zdůvodněte.

22 Relační databáze (DW) (otázka studijního zaměření – 3 body)

Pro potřebu správy rozvrhů jsou definovány následujících uživatelské požadavky:

- Předmět má název, právě jednoho garanta a může být vyučován v LS, ZS nebo obou semestrech.
- Každý předmět může mít 0-N přednášek.
- Každá přednáška může mít 0-N vyučujících.
- Student může být zapsán na přednášku nejvýše jednou.
- Každý předmět může mít 0-N cvičení.
- Každé cvičení má maximální kapacitu a může mít 0-N cvičících.
- Student může být zapsán na cvičení nejvýše jednou.
- Každá přednáška i cvičení má specifikované délku a čas začátku.
- Garanti předmětů, cvičící, přednášející a studenti jsou identifikování pomocí *externího identifikátoru*.

Externí identifikátor, je textový řetězec o maximální délce 45 znaků.

1. Navrhněte databázové relační schéma pro správu rozvrhů podle výše uvedených uživatelských požadavků. Schéma musí splňovat 3NF. Schéma popište pomocí ER diagramu, nebo jako seznam typovaných sloupců v tabulkách s vyznačeným primárním klíčem a cizími klíči.
2. Napište SQL dotaz, který pro vytvořené schéma vrátí pro každý předmět a vyučujícího (přednášející či cvičící), kolik studentů učí.
3. Na straně databáze je třeba logovat vytvoření a rušení zápisů (dotazy INSERT a DELETE) studentů na předměty a cvičení. Navrhněte a popište možné řešení na straně DBMS, bez zásahu do aplikační logiky či změně posílaných dotazů. Popište případné dopady řešení na databázové schéma.

Při řešení můžete použít SQL dialekt běžně známých databází (Oracle, MySQL, PostgreSQL, MSSQL, ...).

23 HTML & CSS (DW) (otázka studijního zaměření – 3 body)

Uvažujme následující fragment HTML kódu:

```
<div class="header">
  Hlavička
  <ul class="navbar">
    <li class="active">Fórum</li>
```

```
    <li>Zápisy</li>
    <li>Přehled</li>
    <li>Nápověda</li>
  </ul>
</div>
<div class="forum">
  <div class="post">
    First
  </div>
  <div class="post">
    Lorem ipsum ..
  </div>
</div>
<div>
  <form action="./create-post.php" method="post">
    <label>
      Post content:
      <textarea name="content" maxlength="280"></textarea>
    </label><br>
    <input type="submit" value="Submit">
  </form>
</div>
<div class="footer">
  <ul>
    <li>Copyright 2022</li>
    <li>Design by ProfDesign</li>
  </ul>
</div>
```

K HTML stránce obsahující zmíněný fragment jsou přilinkované následující CSS definice:

```
li {color:blue}
ul li {color: red;}
.navbar li {color: black;}
.header .navbar li {color: green;}
#container li {color: violet;}
li .class {color:brown}
```

Relevantní úryvek z PHP skriptu "create-post.php"pro validaci dat z formuláře:

```
function isFormPostValid() {
  return isset($_POST['content']) && strlen($_POST['content']) <= 280;
}
```

1. Vysvětlete princip určení specificity CSS selektorů (na čem specificita záleží, jak konkrétně se počítá). Který/které z výše uvedených CSS selektorů má nejvyšší specificitu?
2. Jakou barvu bude mít text "Fórum"v hlavičce ?
3. Na straně serveru jsou data z formuláře validována pomocí uvedeného PHP skriptu. Bylo by možné validaci na straně serveru vynechat? Odpověď stručně zdůvodněte.

24 API & Skriptování (DW) (otázka studijního zaměření – 3 body)

1. Navrhněte a popište REST API pro správu předmětů, cvičení a zápisu na cvičení z předchozího příkladu. API musí umožňovat
   (a) získat seznam předmětů
   (b) vytvořit a zrušit předmět, získat detail předmětu
   (c) získat seznam cvičení předmětu
   (d) vytvořit a zrušit cvičení k předmětu

(e) získat zapsané studenty na jednotlivá cvičení

(f) vytvořit a zrušit zápis studenta na cvičení

API musí používat JSON pro sdílení dat.

2. Napište JavaScript kód, který z daného REST API získá seznam cvičení a následně smaže všechny s nulovou kapacitou. Pokud v řešení použijete funkce poskytnuté prostředím (prohlížeč), krátce popište jejich význam (zejména pokud si nejste jisti přesným názvem). V kódu není třeba extenzivně ošetřovat chybové stavy. Na drobné syntaktické chyby nebude při hodnocení brán zřetel.

3. V rámci implementace vznikl následující fragment PHP kódu:

```
<?php
  $result = $mysqli->query('SELECT * FROM enrollment_view WHERE id="' . $_GET['enrollment'] + '"');
  while ($row = $result->fetch_assoc()) {
    echo('<li>' . $row['subject'] . ':' . $row['student'] . '</li>');
  }
?>
```

Identifikujte a popište alespoň dva závažné bezpečnostní nedostatky v uvedeném PHP kódu.

25 XML (SI) (otázka studijního zaměření – 3 body)

Uvažujte evidenci poslaneckých klubů v Poslanecké sněmovně Parlamentu České republiky. Existuje alespoň jeden poslanecký klub. Pro každý poslanecký klub je evidován jeho název a poslanci, kteří do něj patří. Právě jeden z poslanců v poslaneckém klubu je jeho předsedou. Každý poslanecký klub má minimálně 5 poslanců. Každý poslanec patří právě do jednoho klubu. Pro každého poslance je evidováno jeho jméno rozdělené na křestní jméno a příjmení a politická strana, do které patří. Pro politické strany jsou evidovány jejich názvy.

Vytvořte XML schéma v jazyku XML Schema popisující reprezentaci seznamu poslaneckých klubů dle výše uvedeného zadání v XML. K XML schématu napište také příklad validního XML dokumentu (s alespoň jedním poslaneckým klubem s alespoň jedním poslancem). Napište XPath výraz nad touto XML strukturou, který vrátí klub, jehož předseda je poslanec z politické strany s názvem "Normální politická strana".

26 Návrhové vzory (SI) (otázka studijního zaměření – 3 body)

Uvažujte hlasování poslanců o zákonu. Na začátku je předložen návrh zákona a poslanci k němu mohou podávat pozměňovací návrhy. Každý pozměňovací návrh znamená editaci textu aktuálního znění zákona. Podané pozměňovací návrhy jsou seřazeny do seznamu. Poslanci hlasují o jednotlivých pozměňovacích návrzích tak, jak jsou uvedeny v seznamu.

Vznikl požadavek, aby měli poslanci k dispozici obrazovky, na nichž je aktuální text návrhu zákona zobrazen. V každé chvíli je zobrazen aktuální text návrhu zákona a pozměňovací návrh, o kterém budou poslanci hlasovat v dalším hlasování. Na začátku se jedná o text předloženého návrhu zákona a první pozměňovací návrh. Pokud je pozměňovací návrh schválen, je zobrazen změněný text návrhu zákona a další pozměňovací návrh k hlasování. Pokud pozměňovací návrh není schválen, zobrazený text návrhu zákona zůstává ve své podobě a je zobrazen další pozměňovací návrh k hlasování. Kdykoliv může jakýkoliv poslanec zpochybnit jakékoliv již proběhlé hlasování. Pokud tak učiní, je na obrazovce zobrazen text návrhu zákona v podobě před zpochybněným hlasováním a je opakováno jak zpochybněné, tak všechna po něm následující hlasování.

Na tomto případě vysvětlete návrhový vzor Memento, pomocí kterého navrhněte objektový model kódu pro zobrazování návrhu zákona. Objektový model můžete navrhnout buď v podobě kódu v programovacím jazyku C++, C# nebo Java, nebo v podobě UML diagramu. Návrh doplňte v případě potřeby slovním komentářem.

27 Systémy pro správu verzí (SI) (otázka studijního zaměření – 3 body)

Popište typický proces (workflow) využití distribuovaných verzovacích systémů (například git) při vývoji software v rámci většího týmu. Uveďte příklady běžných situací a problémů, na které můžete narazit při práci s takovým verzovacím systémem, a způsoby jejich řešení. Soustřeďte se především na práci s větvemi, slučování změn provedených nezávisle různými vývojáři z jednoho týmu, a způsoby propagace změn do vzdálených repozitářů (vzhledem k tomu jestli máte či nemáte oprávnění zápisu do toho repozitáře).