Bakalářské zkoušky (příklady otázek z informatiky)

jaro 2020

1 Automaty a gramatiky (3 body)

1. Zformulujte Kleeneho větu. Definujte všechny použité pojmy.
2. Dokažte, že je-li L regulární jazyk, potom jazyk

$$L' = \{w \mid \text{každý prefix } w \text{ liché délky leží v } L\}$$

je také regulární jazyk. (Nápověda: Ukažte, že doplněk jazyka L' je regulární.)

2 Toky v sítích (3 body)

1. Definujte problém hledání maximálního toku v síti.
2. Definujte zlepšující cestu a charakterizujte pomocí ní maximální tok v síti.
3. Zaručuje výběr *nejkratší* zlepšující cesty konečnost a polynomiální časovou složitost pro algoritmus zlepšující cesty (Fordův-Fulkersonův) ve všech případech?
4. V síti jsou navíc zadány maximální kapacity průtoku pro vrcholy. Převeďte problém hledání maximálního toku v síti, kde musí platit i omezení pro průtoky vrcholy na problém hledání maximálního toku v síti bez omezení na vrcholy.

3 Databáze (3 body)

1. Uvažujte transakční rozvrh $T_{12} = X_1(A), W_1(A), U_1(A), X_2(A), R_2(A), U_2(A), S_1(B), R_1(B), U_1(B)$. V zápisu rozvrhu X_i, resp. S_i znamená požadavek na exkluzivní, resp. sdílený zámek i-té transakce, U_i je odemčení. R_i a W_i odpovídá čtení a zápisu proměnné v i-té transakci. Odpovídá rozvrh T_{12} požadavkům na dvoufázový (2PL) uzamykací protokol? Pokud ne, opravte jej tak, aby požadavkům odpovídal a pokud možno nedošlo ke změně vzájemného pořadí všech uvedených čtení a zápisů.
2. Nad tabulkou $Teplota(\underline{Rok, Mesic, Den}, Stupne)$, obsahující data za roky 2010 až 2019, napište dotaz, který zjistí, ve kterých měsících roku 2019 teplota klesla pod nulu alespoň dva dny za sebou. (Situace, kdy teplota klesla pod nulu poslední den jednoho a první den následujícího měsíce neřešte.)

4 Objektově orientované programování a vlákna (3 body)

1. V kontextu níže uvedeného programu v jazyce C# vysvětlete, jak se na úrovni výsledného strojového kódu bude lišit volání nevirtuální metody `f` a virtuální metody `m` v metodě `Print`. Jakým způsobem se za runtime pozná, že se v uvedeném programu jako `x.m` má v metodě `Print` zavolat kód `B` nebo `D`?

```
class X {
    public void f() { A }
    public virtual void m() { B }
}

class Y : X {
    public new void f() { C }
    public override void m() { D }
```

```
}

class Program {
    static void Main() {
        Print(new Y());
    }
    static void Print(X x) {
        x.f();
        x.m();
    }
}
```

(Pokud preferujete, příklad můžete řešit také pro analogickou situaci v jazyce C++.)

2. Napište v jazyce C++, C#, nebo Java, jak by měla vypadat deklarace funkce, která má na místě setřídit pole nějakých objektů `T` (pole je jedním z parametrů funkce) dle nějakého obecného kritéria, které chceme být schopni předat funkci jako parametr. Napište také deklaraci všech potřebných typů.

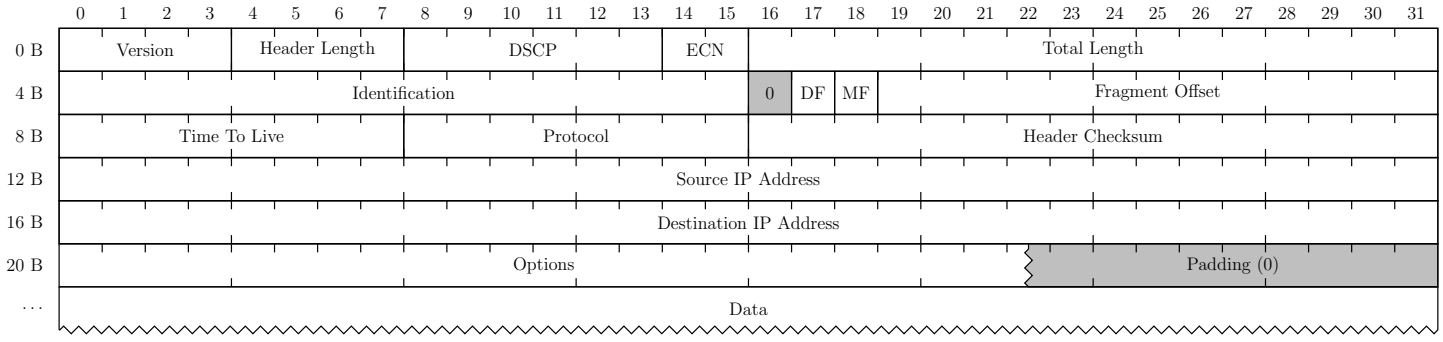
   Napište, jak by se daná funkce volala pro pole níže uvedených typů `Person`, pokud bychom ho jednou chtěli setřídit podle příjmení osob v poli a podruhé podle jejich věku.

```
class Person {
    public string FirstName;
    public string LastName;
    public int Age;
}
```

3. Napište v jazyce C++, C#, nebo Java implementaci funkce `MergeSort`, která dostane 3 parametry – pole nějakých objektů `T`, popis třídícího kritéria (dle řešení části 2), a celé číslo `N`, které je mocninou 2. Funkce má provést paralelní setřídění vstupního pole algoritmem *merge sort* podle zadaného kritéria v právě `N` vláknech (poslední fáze mergování `N` zbylých částí pole může probíhat již v menším počtu vláken než `N`). V řešení můžete používat pomocné pole. V řešení není třeba průběžně zpracovávat výsledky či hledat jiné sofistikované optimalizace, ani není třeba odladit veškeré okrajové případy algoritmu *merge sort* – v této otázce se soustřeďte na korektní vyhodnocování třídícího kritéria a na koncepčně správné vyváření vláken a jejich vzájemnou spolupráci/kooperaci.

5 Architektura počítačů (3 body)

Obrázek 1 znázorňuje strukturu hlavičky IP paketu. Každý řádek reprezentuje jedno 32-bitové slovo, které se skládá ze čtyř bajtů. Čísla na levé straně označují offset v bajtech od začátku paketu. Jednotlivé bity slova jsou číslovány od 0 do 31, přičemž **nejvíce významný bit** (MSb) je v tomto případě označený **číslem 0**. Vícebajtová pole jsou uložena v uspořádání **big endian**, číselná pole obsahují celá nezáporná čísla (unsigned integer). V poli **Header Length** je uložena délka hlavičky paketu jako počet 32-bitových slov. V poli **Total Length** je uložena celková délka paketu (včetně hlavičky) v bajtech. Pokud je nějaký paket příliš dlouhý, může být rozdělen do více částí – fragmentů. Fragmenty jsou také IP pakety (mají stejný formát hlavičky), u nichž je navíc nutné interpretovat pole **Fragment Offset**, ve kterém je uložen offset datové části fragmentu v datové části původního (nefragmentovaného) paketu. Fragment offset je vždy zarovnán na 8 bajtů a v hlavičce je uložen jako počet 8-bajtových bloků od začátku dat. Část **Options** má proměnnou délku a nemusí být vůbec přítomna, protože obsahuje nepovinná pole hlavičky. Část **Padding** má rovněž proměnnou délku a doplňuje hlavičku nulami tak, aby končila na hranici 32 bitů (4 bajtů). Část **Data** obsahuje data paketu, jejichž formát závisí na protokolu.



Obrázek 1: Struktura hlavičky IP paketu

Část A

Na základě informací o struktuře hlavičky IP paketu zodpovězte následující otázky:

1. Jaká je minimální a maximální možná délka hlavičky paketu v bajtech?
2. Jaká je minimální a maximální možná délka celého paketu v bajtech?
3. Jaký je maximální offset fragmentu v bajtech a jakým číslem by byl v poli **Fragment Offset** reprezentován?
4. Může být fragment, který má v poli **Fragment Offset** uvedenou maximální možnou hodnotu, považován za platný? Uvažte, zda by mohl existovat platný nefragmentovaný paket, ze kterého by fragment vznikl. Odpověď zdůvodněte.

Část B

Obrázek 2 znázorňuje výpis obsahu části paměti routeru, který zpracovává IP pakety (všechny hodnoty jsou uvedeny v šestnáctkové soustavě). Na adrese `0xCAFE0010` začíná hlavička IP paketu, jejíž struktura je znázorněna na obrázku 1.

1. Na jaké adrese začíná datová část paketu a jak je dlouhá (v bajtech)?
2. Obsahuje hlavička paketu nepovinnou část (Options) a padding? Pokud ano, jak dlouhá je tato část (včetně paddingu)?

Adresa	*Obsah paměti*							
0xCAFE0000	BE	EF	7C	B2	7D	45	FE	9D
0xCAFE0008	DC	02	8E	30	CA	EA	08	00
0xCAFE0010	45	00	00	34	B1	BF	40	00
0xCAFE0018	36	06	06	FA	04	1F	C6	3E
0xCAFE0020	C0	A8	01	05	01	BB	9A	6C
0xCAFE0028	82	6A	E0	E4	5E	31	44	E4
0xCAFE0030	80	10	00	73	43	2B	00	00
0xCAFE0038	01	01	08	0A	F4	DF	51	5B
0xCAFE0040	1C	30	A2	1C	FE	31	1F	C0

Obrázek 2: Výpis paměti obsahující IP paket

Část C

Předpokládejte, že v routeru je použit jednoduchý 32-bitový RISC procesor s load/store architekturou. Procesor má 16 obecných (general-purpose) 32-bitových registrů označených `$r0`–`$r15`, přičemž registr `$r0` vždy obsahuje hodnotu `0`. Tabulka na obrázku 3 poskytuje stručný přehled základních instrukcí procesoru. V popisu instrukcí se používá *imm32* pro 32-bitové konstanty a *label* pro symbolické návěští nahrazující adresu v kódu. Paměť je adresována obvyklým způsobem (po bajtech), ale kvůli jednoduchosti architektury lze přistupovat **pouze k celým 32-bitovým slovům** na zarovnaných adresách, tj. na adresách **dělitelných 4**. Při čtení/ukládání slov do paměti procesor pracuje v režimu **little-endian**.

Přístup do paměti		
lw $rd, *imm32* ($rs)	Čtení 32-bitového slova do registru	R[rd] = Memory[R[rs] + *imm32*]
sw $rs, *imm32* ($rd)	Zápis 32-bitového slova do paměti	Memory[R[rd] + *imm32*] = R[rs]
Aritmetické a logické operace		
add/sub $rd, $rs, $rt	Sčítání/odčítání registrů	R[rd] = R[rs] ± R[rt]
addi/subi $rd, $rs, *imm32*	Sčítání/odčítání registru a konstanty	R[rd] = R[rs] ± *imm32*
and/or/xor $rd, $rs, $rt	Bitové and/or/xor s registrem	R[rd] = R[rs] *op* R[rt]
andi/ori/xori $rd, $rs, *imm32*	Bitové and/or/xor s konstantou	R[rd] = R[rs] *op* *imm32*
sll/slr/sra $rd, $rs, $rt	Posuny, logický vlevo/vpravo, aritmetický vpravo	R[rd] = R[rs] *op* (R[rt] **mod** 32)
slli/slri/srai $rd, $rs, *imm32*	Posuny o konstantu, logický vlevo/vpravo, aritmetický vpravo	R[rd] = R[rs] *op* (*imm32* **mod** 32)
li $rd, *imm32*	Načtení konstanty do registru	R[rd] = *imm32*
move $rd, $rs	Přesun (kopie) hodnoty mezi registry	R[rd] = R[rs]
Porovnání a podmíněné skoky		
slt $rd, $rs, $rt	Znaménkové *set on less than* (<) mezi registry	**if** (R[rs] $<_{signed}$ R[rt]) **then** R[rd] = 1 **else** R[rd] = 0
sltu $rd, $rs, $rt	Neznaménkové *set on less than* (<) mezi registry	**if** (R[rs] $<_{unsigned}$ R[rt]) **then** R[rd] = 1 **else** R[rd] = 0
slti $rd, $rs, *imm32*	Znaménkové *set on less than* (<) s konstantou	**if** (R[rs] $<_{signed}$ *imm32*) **then** R[rd] = 1 **else** R[rd] = 0
sltiu $rd, $rs, *imm32*	Neznaménkové *set on less than* (<) s konstantou	**if** (R[rs] $<_{unsigned}$ *imm32*) **then** R[rd] = 1 **else** R[rd] = 0
beq $rs, $rt, *label*	Větvení (skok na adresu/návěští) při rovnosti registrů	**if** (R[rs] == R[rt]) **then** PC = *label* (adresa)
bne $rs, $rt, *label*	Větvení (skok na adresu/návěští) při nerovnosti registrů	**if** (R[rs] != R[rt]) **then** PC = *label* (adresa)
Nepodmíněné skoky		
j *label*	Skok na adresu/návěští (*label*)	PC = *label* (adresa)
jr $rs	Skok na adresu v registru	PC = R[rs]

Obrázek 3: Přehled instrukcí procesoru

S využitím informací o struktuře hlavičky IP paketu a při dodržení omezení daných architekturou procesoru napište s pomocí popsané instrukční sady fragment kódu, který v hlavičce paketu sníží hodnotu pole **Time To Live** (TTL) o `1`, ovšem **pouze pokud je nenulová** (aby nedošlo k podtečení). Předpokládejte, že adresa paketu je v registru `$r1` a že je zarovnána (aligned) na 4 B.

6 Architektura TCP/IP (3 body)

Váš nástroj pro sledování provozu sítě zachytil následující paket:

```
0000   00 1f 1f c0 ca 33 54 e1 ad 15 39 92 08 00 45 00   .....3T...9...E.
0010   00 5c 21 f2 40 00 40 06 df f9 c0 a8 00 10 5f d8   .\!.@.@........_.
0020   18 20 8f e6 00 50 c7 f5 f6 7d f8 77 22 8c 80 18   . ...P...}.w"...
0030   01 f6 34 3d 00 00 01 01 08 0a 30 c9 1c 8b 9b 1a   ..4=......0.....
0040   19 29 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 30   .)GET / HTTP/1.0
0050   0d 0a 48 6f 73 74 3a 20 77 77 77 2e 61 70 61 63   ..Host: www.apac
0060   68 65 2e 6f 72 67 0d 0a 0d 0a                     he.org....
```

1. Pokud víte, že provoz probíhal IP protokolem ze zdrojové adresy 192.168.0.16 (formát IP paketu je v předchozí otázce), vyznačte ve výpisu data linkové vrstvy (link layer) a síťové vrstvy (internet layer). Co za informace pravděpodobně obsahují data linkové vrstvy ?
2. Z obsahu paketu odhadněte také pozici dat transportní vrstvy (transport layer) a aplikační vrstvy (application layer) a svůj odhad vysvětlete (k přesnému určení pozice by byla potřeba znalost dalších standardních formátů paketů, informovaný odhad však v tomto případě může fungovat také dobře). Jak velká (v bajtech) jsou data jednotlivých vyznačených vrstev (všech čtyř) ?
3. Odhadněte dále, jaké aplikace spolu tímto paketem komunikují a jaké další protokoly kromě již zmíněného IP protokolu používají. Váš odhad zdůvodněte.
4. Pokud by si komunikující aplikace potřebovaly předat větší blok dat, než jaký lze po cestě mezi odesilatelem a příjemcem přenést v jednom kuse (tedy pokud by například odesilatel požádal o odeslání bloku dat o velikosti 1 MB systémovým voláním `write` na příslušném spojení), kde by v uvedeném schématu došlo k jeho rozdělení a spojení ?

7 Morfologická, syntaktická a sémantická analýza přirozeného jazyka (otázka studijního zaměření – 3 body)

1. Vysvětlete rozdíl mezi stemmingem a lematizací slov přirozeného jazyka a uveďte jazykový jev, kvůli kterému není používání stemmingu vhodné pro češtinu.
2. Co znamená pojem Universal Dependencies a proč jsou důležité?
3. Vysvětlete hlavní myšlenku algoritmu zásoby sdílených znalostí (Stock of Shared Knowledge).

8 Základní formalismy pro popis přirozeného jazyka (otázka studijního zaměření – 3 body)

1. Definujte sestavu rysů (feature structure) jako základní datovou strukturu unifikačních gramatik.
2. Vysvětlete operaci unifikace sestav rysů a zdůvodněte, proč je nutné používat typované sestavy rysů.
3. Stručně uveďte základní vlastnosti formalismu HPSG (Head Driven Phrase Structure Grammar).

9 Základy teorie informace (otázka studijního zaměření – 3 body)

Házíme hrací kostkou a hozené číslo z množiny $\{1, 2, 3, 4, 5, 6\}$ interpretujeme jako hodnotu náhodné proměnné X. Předpokládejme, že X má uniformní rozdělení. Dále uvažujme náhodnou proměnnou Y s hodnotami *sudé/liché* a náhodnou proměnnou Z s hodnotami *true* (pokud padne číslo větší než 4) nebo *false* (pokud nepadne číslo větší než 4). Obory hodnot náhodných proměnných jsou shrnuty v tabulce

náhodná proměnná	**hodnoty**
X	$\{1, 2, 3, 4, 5, 6\}$
Y	$\{sudé, liché\}$
Z	$\{true, false\}$

1. Která z proměnných X Y Z má největší entropii? Která má nejmenší entropii? Odpověď přesně zdůvodněte.
2. Vypočtěte podmíněnou entropii $\mathrm{H}(X|Z)$. Uveďte postup výpočtu.

10 Barvy v počítačové grafice (otázka studijního zaměření – 3 body)

1. Jak barvy vnímá lidské oko? Uveďte alespoň přibližný koncept.
2. Jak barvy zobrazuje počítačová technika? Popište systémy pro ukládání barev na počítači (barevné prostory).
3. Napište příklad barevného systému vhodného pro intuitivní zadávání barvy laickým uživatelem.

11 Distribuované sledování paprsku (otázka studijního zaměření – 3 body)

Tyto techniky se také nazývají "Monte-Carlo ray tracing" nebo "Monte-Carlo integrace".

1. Vyjmenujte několik příkladů, kde je užitečné do rekurzivního sledování paprsku zapojit stochastický výpočet (Monte-Carlo integraci).
2. Vyberte si jedno konkrétní použití Monte-Carlo a popište ho detailně. Které veličina se integruje? Který nedostatek původního přístupu se odstraňuje nebo potlačuje?
3. Navrhněte vzorkování (sampling), které by se dalo dobře použít v předchozím příkladu. Stačí popsat metodu vzorkování rámcově nebo nakreslit obrázek. Šlo by toto vzorkování implementovat adaptivně?

12 Objemové textury (otázka studijního zaměření – 3 body)

1. Vysvětlete princip objemové (3D) textury použité v ray-tracingu. Jak se její použití liší od použití 2D textury?
2. Navrhněte přírodní materiál, který by se dal úspěšně simulovat technikou 3D textury.

3. Dala by se procedurální objemová textura použít i v realtime GPU grafice? Naznačte stručně princip.

13 C++ traits a policies (otázka studijního zaměření – 3 body)

1. Stručně (2-3 věty) vysvětlete pojem **trait** v kontextu jazyka C++. Uveďte jednoduchý příklad.
2. Uvažme následující šablonovanou třídu `Matrix<T>`, která reprezentuje matici. Navrhněte úpravu této třídy tak, aby akceptovala politiku (policy), která bude ovlivňovat způsob uložení dat matice do interního pole (vektoru). Odpovídajícím způsobem upravte také tělo metody `at()`. Jako příklad implementujte dvě politiky – jednu, která ukládá data po řádcích (jako původní implementace matice), a druhou, která ukládá data po sloupcích (tedy transponovaně).

```
template<typename T> Matrix {
private:
    std::size_t mWidth, mHeight;
    std::vector<T> mData;
public:
    T& at(std::size_t i, std::size_t j) {
        return mData[j*mWidth + i];
    }
    ...
};
```

14 Návrhový vzor Decorator (otázka studijního zaměření – 3 body)

1. Stručně (v pár větách) vysvětlete hlavní podstatu návrhového vzoru Decorator.
2. Mějme rozhraní `ILogger`, které definuje metody `error()`, `warning()` a `info()`. Každá metoda má právě jeden argument – řetězec obsahující zprávu k zalogování. Předpokládejme, že existují různé implementace rozhraní ILogger, které zprávy ukládají do různých úložišť (do souboru, do databáze, ...). V libovolném jazyce z množiny C++/C#/Java navrhněte a implementujte dekorátor, který k logovaným zprávám přidá pevně daný prefix (např. jméno uživatele).
3. Navrhněte úpravu vaši implementace z předchozího bodu tak, aby vkládaný prefix mohl být dynamický, tedy vypočítaný individuálně pro každou logovanou zprávu (např. aktuální datum a čas při vložení zprávy). Samotná implementace dekorátoru by měla být obecná pro libovolné dynamické prefixy. Ve vašem řešení stačí dostatečně jasně uvést změny oproti implementaci z předchozího bodu (není třeba popisovat celou implementaci znovu).

15 Autentizace ve webových aplikacích (otázka studijního zaměření – 3 body)

Mějme webovou aplikaci, která autentizuje uživatele kombinací přihlašovacích údajů – přihlašovacího jména (loginu) a hesla.

1. Navrhněte způsob, jak bezpečně uložit hesla do databáze, aby je nemohl jednoduše přečíst např. administrátor, a zároveň aby bylo možné je ověřovat při přihlašování.
2. Jak musí být aplikace zabezpečena, aby nemohlo dojít o odposlechnutí hesla při přihlašování uživatele (pomocí klasického útoku man-in-the-middle). Stručně vysvětlete, na jakém algoritmickém principu vámi navržené zabezpečení stojí (stačí např. názvy algoritmů).
3. Co je to **digitální certifikát** webového serveru a jakým způsobem ověří prohlížeč/uživatel jeho platnost?

16 Spouštění procesů a volací konvence (otázka studijního zaměření – 3 body)

1. Stručně vysvětlete, proč je nutné jako součást ABI (Application Binary Interface) definovat volací konvence a co volací konvence popisují.
2. Uvažujte procesor schopný relativní adresace vzhledem k obsahu běžných registrů (tedy operandy instrukcí mohou být například výrazy jako `[REG+CON]` pro obsah paměti na adrese určené přičtením konstanty `CON` k obsahu registru `REG`). Navrhněte pro takový procesor volací konvence používající zásobník pro předávání parametrů, zásobník musí dovolovat inspekci za běhu programu (stack frame walk).
3. Načrtněte stav zásobníku při použití vaší volací konvence v situaci, kdy se program nachází v těle funkce `f (1, 2)` volané z funkce `g (3, 4)` volané z funkce `main ()`.

Pokud pro svůj návrh potřebujete libovolné další technické detaily, zvolte je podle vlastního uvážení.

17 Správa verzí (otázka studijního zaměření – 3 body)

Uvažujte systém pro správu verzí `git`.

1. V repozitáři jste spustili následující příkazy a obdrželi následující výstup:

   ```
   > git branch
   * master
   > git status
   On branch master
   Your branch is up to date with 'origin/master'.
   > _
   ```

   Co znamená `master` a jaký je vztah mezi `master` a `origin/master` ?

2. V témže repozitáři jste dále spustili následující příkazy a obdrželi následující výstup:

   ```
   > git branch --all
   * master
     remotes/origin/brand-new-feature
     remotes/origin/master
   > git checkout brand-new-feature
   Branch 'brand-new-feature' set up to track remote branch 'brand-new-feature' from 'origin'.
   Switched to a new branch 'brand-new-feature'
   > _
   ```

   Co znamená `remotes/origin/brand-new-feature` a co znamená "track remote branch" ?

3. Popište, jak modifikace v `brand-new-feature` po dokončení integrujete do `master`.

18 Zasílání zpráv, point-to-point, publish-subscribe (otázka studijního zaměření – 3 body)

1. Vysvětlete rozdíl mezi modely "point-to-point" a "publish-subscribe" pro zasílání zpráv. Pro oba modely navrhněte rozhraní sloužící k odeslání a příjmu zprávy a popište funkci rozhraní včetně významu použitých argumentů. Očekává se návrh vhodný pro běžný procedurální nebo objektový imperativní programovací jazyk, syntaxe může být pouze přibližná.

2. Zvolte jeden z uvedených modelů a načrtněte, jak by se pomocí rozhraní z předchozího bodu dala implementovat aplikace pro diskuzní skupiny, jejíž uživatelé se mohou přihlašovat do diskuzních skupin a rozesílat zprávy všem přihlášeným členům skupiny.