Bakalářské zkoušky (příklady otázek z informatiky)

podzim 2020

1 Automaty a gramatiky (3 body)

1. Formulujte Iterační (pumping) lemma pro bezkontextové jazyky.
2. Zařaďte jazyk $L = \{a^2 b^i c^i \mid i \in \mathbb{N}_0\}$ (slovo začíná 'aa', po něm libovolný počet 'b' následovaný stejným počtem 'c') do Chomského hierarchie. Tj. určete nejmenší třídu (nejvyšší typ) Chomského hierarchie, do které L náleží, a dokažte, že náleží do Vámi uvedené třídy a nepatří do třídy menší.

2 Topologické uspořádání (3 body)

1. Definujte topologické uspořádání orientovaného grafu. Napište charakteristiku grafů, které jdou topologicky uspořádat.
2. Popište algoritmus konstruující topologické uspořádání orientovaného grafu.
3. V daném orientovaném grafu G, který lze topologicky uspořádat, chceme *efektivně* najít pro dva dané vrcholy u a v všechny vrcholy, které leží na nějaké orientované cestě z u do v. Stručně rozeberte složitost vašeho algoritmu (v závislosti na počtu vrcholů n a počtu hran m grafu G).

3 Databáze (3 body)

1. Které dvojice databázových operací v transakčním rozvrhu jsou konfliktní? Vysvětlete, kdy jsou dva transakční rozvrhy stejné množiny transakcí konfliktově ekvivalentní.
2. Uvažujte transakce T_1: $R(Y)$ $W(Y)$ $W(X)$ a T_2: $R(Y)$ $W(Y)$ $R(X)$ a rozvrh S: $R_1(Y)$ $W_1(Y)$ $R_2(Y)$ $W_2(Y)$ $R_2(X)$ $W_1(X)$ $COMMIT_2$ $COMMIT_1$. Je rozvrh S konfliktově uspořádatelný (conflict–serializable)? Zdůvodněte a případně navrhněte opravu na konfliktově serializovatelný rozvrh.
3. Bude relace *R(A,B,C), kde* F=$\{A \to B,\ B \to C,\ C \to A\}$ z hlediska normálních forem vhodně navržená? Pokud ne, jak by bylo vhodné návrh změnit a proč?

4 Objektový návrh (3 body)

Předpokládejte, že programujeme základ frameworku pro webové aplikace – níže uvedené fragmenty kódu jsou zapsány v jazyce C#, nicméně vaše řešení můžete psát v C#, Javě, nebo C++. Pro zjednodušení ve své implementaci neošetřujte chybové stavy, neočekávané hodnoty parametrů, apod. Ve svém řešení si můžete dopsat libovolné potřebné pomocné metody a typy nad rámec požadavků dané části otázky.

1. Připravili jsme třídu `MasterController1`, která bude rozdělovat zpracování příchozích požadavků dle jejich `id`:

   ```
   class MasterController1 { public string HandleReq(string id) {  } }
   ```

   Každý druh požadavku (unikátního `id`) budeme zpracovávat v jiné nezávislé třídě tzv. controlleru požadavku. Napište implementaci třídy `GreetingsController1`, která řeší požadavky s ID „`welcome`“, a to vrácením konstantního řetězce „`Hello`“, a dále třídy `SumController1` pro požadavky „`sum`“, která jako textový řetězec vrací vždy součet čísel 10 a 20. Předpokládejte, že instance těchto tříd budou vždy bezstavové, ale v budoucnu mohou dělat komplexnější zpracování požadavku, než je výše popsáno.

   Doplňte implementaci `MasterController1` tak, aby u ní bylo možno registrovat dvojice „ID požadavku“ a „instance požadavek řešící třídy `C`“, a to následujícím způsobem:

```
var mc = new MasterController1();
mc.RegisterController("welcome", new GreetingsController1());
mc.RegisterController("sum", new SumController1());
```

Napište tělo metody `HandleReq` na `MasterController1` tak, aby zařídilo zpracování požadavku zaregistrovaným controllerem dle `id` požadavku, a vrátilo výsledek jeho zpracovávací metody.

2. Máme rozšířenou verzi V2 třídy:

   ```
   class MasterController2 { public string HandleReq(string id, string[] args) {  } }
   ```

   V nové verzi předpokládáme, že controllery řešící jednotlivé požadavky jsou stavové – zpracování každého požadavku probíhá postupně, kdy se controlleru požadavku postupně předávají různé parametry zpracování, a ten na jejich základě vygeneruje výsledek požadavku jako jeden textový řetězec.

   Doplňte implementaci `MasterController2` tak, aby dle `id` požadavku vybranému controlleru postupně po jednom předala parametry z pole `args`, a poté zavolala metodu, která vrátí výsledek zpracování. Dále napište implementaci verze V2 třídy `GreetingsController2`, která vrátí text „`Hello`“ následovaný parametry oddělenými mezerou; a verzi V2 implementace `SumController2`, která předané parametry chápe jako celá čísla, a jako výsledek zpracování vrátí jejich celočíselný součet v textové reprezentaci. Způsob registrace controllerů a jim náležejícím ID požadavků zůstane stejný jako v části 1 – tedy následující kód využívající proměnnou `mc` zkonstruovanou dle části 1, ale s verzemi V2 všech tříd:

   ```
   Ř1: Console.WriteLine(mc.HandleReq("welcome", new[] { "Princess", "Consuela", "Banana", "Hammock" }));
   Ř2: Console.WriteLine(mc.HandleReq("sum", new[] { "1", "2", "39" }));
   Ř3: Console.WriteLine(mc.HandleReq("sum", new[] { "1000", "500" }));
   ```

   vypíše (všimněte si, že na posledním řádku je hodnota 1542 místo 1500, jelikož třída `MasterController2` využívá stejnou zaregistrovanou instanci `SumController2` pro každý požadavek s ID „`sum`“):

   ```
   Hello Princess Consuela Banana Hammock
   42
   1542
   ```

3. Upravte řešení z části 2 tak, aby třída `MasterController2` pro každý požadavek nějakého ID používala vždy novou čistě inicializovanou instanci třídy zaregistrované pro dané ID. Tedy aby např. provedení výše uvedeného kódu řádku Ř2 použila `MasterController2` jednu instanci `SumController2` a ta vrátila 42, a na řádku Ř3 se použila jiná nová instance `SumController2` a ta pak tedy vrátila 1500 místo 1542.

   Při registraci dvojice „ID požadavku“, a „požadavek řešící třída `C`“ budete muset třídě `MasterController2` místo instance typu `C` nějakým způsobem předat „návod“, resp. „nástroj“, jak má `MasterController2` instance `C` konstruovat.

5 Architektura počítačů a systém souborů FAT (3 body)

Pokusíme se pracovat se starším souborovým systémem FAT, konkrétně v jeho verzi FAT12 (používal se na disketách). Tento souborový systém ukládá data do bloků nazývaných *clustery*, ty jsou identifikovány svým pořadovým číslem. Vedle prostoru pro data ukládá souborový systém na disk ještě *tabulku FAT*, která obsahuje jednu položku pro každý cluster (jde o pole indexované číslem clusteru, zde od 0). Položka říká, zda je cluster obsazený souborem, a pokud ano, který další cluster v souboru následuje.

Ve formátu FAT12 má jedna položka tabulky FAT délku 12 bitů, takže dvě položky ve FAT12 jsou uložené ve třech bytech. Pokud tyto byty jsou šestnáctkově UV, WX, YZ, pak položky jsou XUV a YZW. Možné hodnoty položek jsou:

Položka	Význam
000	Volný cluster
002-fef	Použitý cluster
ff0-fff	Použitý cluster, konec souboru

Toto je hexadecimální výpis začátku tabulky FAT (zbytek tabulky obsahuje samé 0):

```
000200   f0   ff   ff   00   40   00   05   60   00   07   80   00   09   a0   00   0b   ....@..`........
000210   c0   00   0d   10   01   ff   0f   01   ff   2f   01   13   40   01   15   60   ........../..@..`
000220   01   17   80   01   19   a0   01   1b   c0   01   1d   e0   01   1f   00   02   ................
000230   21   20   02   ff   0f   00   00   00   00   00   00   00   00   00   00   00   ! ..............
```

Kořenový adresář má jen 3 položky zachycené v následujícím hexadecimálním výpisu:

```
002600   46   4f   4c   44   45   52   20   20   20   20   20   20   00   00   00   00   FOLDER      ....
002610   00   00   00   00   00   00   a0   79   79   4f   03   00   9d   39   00   00   .......yyO...9..
002620   46   49   4c   45   20   20   20   20   54   58   54   10   00   00   00   00   FILE    TXT.....
002630   00   00   00   00   00   00   85   78   1f   51   0e   00   00   00   00   00   .......x.Q......
002640   4c   4f   43   43   4f   4e   46   20   58   4d   4c   20   00   00   00   00   LOCCONF XML ....
002650   00   00   00   00   00   00   dc   aa   6f   4c   0f   00   96   03   00   00   ........oL......
```

Položka adresáře má následující strukturu:

Byty	Obsah		
0-10	Jméno souboru (8B) a přípona (3B)		
11	Atributy souboru (po bitech):	Bit	Význam
		0	read only
		1	hidden
		2	system file
		3	volume label
		4	subdirectory
		5	archive
12-21	Rezervováno		
22-23	Čas poslední změny (složitý formát, není podstatný)		
24-25	Datum poslední změny (složitý formát, není podstatný)		
26-27	Počáteční cluster		
28-31	Velikost souboru v bytech		

Souborový systém FAT ukládá číselné hodnoty jako little endian, bit 0 je LSB.

1. Vypište obsah kořenového adresáře (jména souborů, zda jde o soubor nebo adresář, u souborů velikost v bytech v desítkové soustavě).
2. Jaké clustery musíte načíst, pokud čtete sekvenčně soubor LOCCONF.XML?
3. Kam a co (stačí koncepčně, není třeba přesně vypsat jednotlivé byty) musím zapsat, pokud na konec souboru LOCCONF.XML zapíši nová data o velikosti 1 KiB?

6 Transportní protokoly a jejich vlastnosti (3 body)

Vysvětlete následující pojmy. Stručně také vyjmenujte, čeho jejich prostřednictvím chceme dosáhnout a jaké problémy v rámci jejich zajištění musíme řešit.

1. Proudový přenos (stream-oriented transfer)
2. Spojovaný přenos (connection-oriented transfer)
3. Spolehlivý přenos (reliable transfer)

Navrhněte a pečlivě popište, jak byste tyto vlastnosti zajistili u hypotetického (přesto smysluplně a efektivně fungujícího) protokolu na transportní vrstvě, který má fungovat nad IP protokolem v síťové vrstvě. Inspirovat se můžete existujícími protokoly v architektuře TCP/IP, stejně tak můžete navrhnout vlastní řešení. Konkrétně se věnujte následujícím oblastem:

1. Vytvoření iluze proudového přenosu
2. Navázání nového spojení
3. Zajištění spolehlivosti

7 Optimalizace, mnohostěny (3 body)

1. Uvažte mnohostěn P: $x_1 + x_2 \geq 1$, $x_2 + x_3 \geq 1$, $x_1 + x_3 \geq 1$. Rozhodněte, zda P je plnodimensionální a zda je celočíselný (všechny jeho vrcholy mají celočíselné souřadnice).
2. Pro daný graf $G = (V, E)$ uvažte lineární program $\min \sum_{e \in E} x_e$ t.ž. $\sum_{e \in T} x_e \geq 1$ pro každou kostru T grafu G, $x \geq 0$. Čemu odpovídá celočíselné optimální řešení?
3. Je pro každý graf G mnohostěn z předchozí otázky celočíselný?

8 Aproximační algoritmy (3 body)

1. Zaveďte pojem *aproximační faktor algoritmu*, pro deterministické i pravděpodobnostní algoritmy.
2. Pro problém MAX-E3SAT (SAT, kde každá klauzule obsahuje právě tři literály) popište pravděpodobnostní algoritmus s aproximačním faktorem 7/8.
3. Dokažte, že aproximační faktor algoritmu z předešlé otázky není ani lepší, ani horší než 7/8.

9 Aranžmá přímek (3 body)

1. Definujte *aranžmá přímek v rovině*.
2. Popište (jakýkoliv) způsob reprezentace aranžmá přímek v rovině v počítači.
3. Navrhněte (jakýkoliv) polynomiální algoritmus, který zkonstruuje aranžmá přímek v rovině ze zadané množiny přímek, a odhadněte co nejlépe jeho (asymptotickou) časovou složitost.

   (Při popisu algoritmu je důležité popsat, z jakých dílčích kroků (popř. subalgoritmů) algoritmus sestává, ale není potřeba zacházet do podrobností ohledně implementace takových dílčích kroků.)

10 Kódy (3 body)

1. Pro řetězec w skládající se ze znaků 0 a 1 označme jako $\overline{w}$ řetězec, který z něj vznikne výměnou všech 0 za 1 a naopak, např. $\overline{0010} = 1101$. Pro řetězce w_1 a w_2 označme $w_1 w_2$ jejich zřetězení.

   Nechť $H_0 = 1$ a pro $i \geq 0$ induktivně definujme $H_{i+1} = \{ww, w\overline{w} : w \in H_i\}$.

   Určete parametry (délku, velikost a minimální vzdálenost) kódu H_n v závislosti na $n \in \mathbb{N}$.
2. Hrajeme následující hru: Soupeř si myslí jedno z čísel $\{1, \ldots, m\}$, kde m je mocnina dvojky. My smíme položit m dotazů ve tvaru „Patří číslo, které si myslíš, do následující množiny: … ?“. Soupeř nemusí vždy odpovídat pravdivě, zalhat nám ale může pouze méně než $(m/4)$-krát. Zvítězíme, pokud poté uhodneme myšlené číslo.

   Popište vyhrávající strategii pro tuto hru.

11 Relační databáze, internacionalizace (otázka studijního zaměření – 3 body)

Mějme tabulku `assignments` v relační databázi, která uchovává zadání domácích úkolu pro studenty. Tabulka má následující sloupce:

- `id` (INT) – uměle přiřazený primární klíč
- `name` (VARCHAR) – název úkolu
- `specification` (TEXT) – zadání úkolu v Markdown
- `points` (INT) – počet bodů, které může student nejvýše získat úspěšným vyřešením úkolu
- `deadline` (DATE) – do kdy je možné odevzdávat řešeni

Rozhodnutím děkana začala škola přijímat i zahraniční studenty, a proto je potřeba patřičně upravit software pro zadávání domácích úkolů.

1. Navrhněte úpravu databáze tak, aby bylo možné tentýž úkol vytvořit ve více překladech (např. česky a anglicky). Pro identifikaci jazyka budeme používat dvoupísmenný identifikátor standardu ISO 639-1 (např. čeština je `cs`, angličtina `en`).
2. Napište SQL dotaz, který vrátí stejné výsledky (tj. stejné sloupce a všechny záznamy úkolů), jako by vrátil `SELECT * FROM assignments` provedený nad původní tabulkou, s tím, že lokalizované položky vrátí pouze anglicky (pokud pro daný úkol angličtina chybí, položky budou mít hodnotu `NULL`).
3. Upravte předchozí dotaz tak, aby překládané texty u každé úlohy byly česky (kde je to možné), nebo anglicky (pokud neexistuje český překlad, ale existuje anglický), nebo v libovolném jiném jazyce (pokud neexistuje český ani anglický překlad, ale existuje alespoň jeden libovolný jiný). U záznamů, které nemají žádný jazykový překlad, nechť je hodnota obou položek `NULL`.

Při řešení tolerujeme použití SQL dialektu běžně známých databází (Oracle, MySQL, PostgreSQL, MSSQL, …).

12 Programování na straně klienta, JavaScript (otázka studijního zaměření – 3 body)

Mějme webovou aplikaci s formulářem, ve kterém je textové pole:

```
<input id="query" name="query" type="text" ... >
```

K textovému poli chceme implementovat inteligentní našeptávač v JavaScriptu. Našeptávač dostane pole potenciálních záznamů `candidates` a funkci `isRelevant(input, candidate)`, která porovná řetězec `input` ze vstupu s jednou položkou pole `candidates` a vrátí *true* či *false* podle toho, zda je daný kandidát relevantní k danému vstupu (tj. má být nabídnut našeptávačem).

Triviální implementace nalezení všech vhodných kandidátu by tedy mohla vypadat takto:

```
const input = document.getElementById("query").value;
const suggestions = candidates.filter(candidate => isRelevant(input, candidate));
```

Hlavní problém výše uvedeného kódu spočívá v tom, že výpočet může trvat delší dobu (naše filtrovací funkce je výpočetně náročná a kandidátů může být poměrně mnoho). Funkce `isRelevant` je sice deterministická, avšak velmi složitá, a proto není možné předjímat její výsledky pro různé vstupy.

Dále máte funkce `showSuggestions(suggestions)` a `hideSuggestions()`, které zobrazí, resp. skryjí seznam s nabízenými položkami (detaily výběru již dále neřešíme).

Navrhněte implementaci výše popsaného našeptávače tak, aby se aktualizoval (a zobrazil) seznam kandidátů při každé změně obsahu vstupního pole, avšak nesmíte zablokovat hlavní smyčku zpracování událostí na delší dobu. Nejdůležitější části zapište jako fragmenty JavaScript kódu, v případě potřeby je doplňte komentářem. Na drobné syntaktické chyby a nepřesné názvy vestavěných funkci nebude brán zřetel, pokud bude z kódu zřejmý záměr.

Doplňme ještě dvě technické poznámky: Funkce `showSuggestions` je výpočetně málo náročná a je možné ji volat i opakovaně a průběžné tak aktualizovat nabízený seznam. Pole relevantních kandidátů předané funkci `showSuggestions` nemusí zachovávat pořadí z původního pole `candidates` (funkce `showSuggestions` si kandidáty sama setřídí).

13 REST (otázka studijního zaměření – 3 body)

Chceme implementovat single-page webovou aplikaci (tj. aplikaci implementovanou převážně na straně klienta), která nabídne uživateli zjednodušený unixový terminál přímo v prohlížeči. Terminál dovolí uživateli se přihlásit (pomocí lokálního UNIXového účtu), spustit libovolný příkaz v shellu, a zobrazit jeho výstup v prohlížeči.

Navrhněte RESTové API, které dovolí výše uvedenou aplikaci realizovat. Nezapomeňte, že zadaný příkaz v shellu může spustit celou řadu procesů, jejichž dokončení muže trvat podstatně déle, než jak dlouho je možné držet otevřené jedno HTTP spojení. Do textu popište především všechny endpointy API (jako URL a HTTP metodu) a u každého stručně vysvětlete jeho funkci a případně očekávaná data v požadavku a odpovědi (pokud to není zřejmé). Naopak vůbec nemusíte popisovat konkrétní implementaci vámi navrženého API (je plně postačující, že taková implementace bude ve vhodném jazyce možná).

Navrhněte rozšíření API tak, aby dovolovalo provést `scp` příkaz, jehož vstupní nebo výstupní data dodá nebo odebere prohlížeč – tedy aby mohl uživatel uploadovat soubor z prohlížeče do adresáře, ve kterém se příkaz na serveru vykonává, nebo naopak stáhnout takový soubor ze serveru do prohlížeče.

Zde je pro ilustraci *příklad* popisu endpointu z jiného REST API:

`GET /num/add` – sečte dvě celá čísla `x` a `y`, která dostane jako parametry v query URL; výsledek vrátí textově v decimálním zápisu

14 Dobré uspořádání (3 body)

1. Napište definici dobrého uspořádání (pojem „uspořádání“ definovat nemusíte).
2. Je každé dobré uspořádání lineární? (krátce zdůvodněte)
3. Uveďte příklad nekonečné spočetné množiny A a tří lineárních uspořádání R, S, T na A takových, že R není dobré, S a T jsou dobrá, přitom uspořádané množiny (A, S) a (A, T) nejsou izomorfní.

15 Perfektní párování (3 body)

1. Zformulujte Tutteho větu o existenci perfektního párování.
2. Je pravda, že každý souvislý 4-regulární graf se sudým počtem vrcholů má perfektní párování? Vysvětlete.

16 Kódy (3 body)

1. Pro řetězec w skládající se ze znaků 0 a 1 označme jako $\overline{w}$ řetězec, který z něj vznikne výměnou všech 0 za 1 a naopak, např. $\overline{0010} = 1101$. Pro řetězce w_1 a w_2 označme $w_1 w_2$ jejich zřetězení.

   Nechť $H_0 = 1$ a pro $i \geq 0$ induktivně definujme $H_{i+1} = \{ww, w\overline{w} : w \in H_i\}$.

   Určete parametry (délku, velikost a minimální vzdálenost) kódu H_n v závislosti na $n \in \mathbb{N}$.

2. Hrajeme následující hru: Soupeř si myslí jedno z čísel $\{1, \ldots, m\}$, kde m je mocnina dvojky. My smíme položit m dotazů ve tvaru „Patří číslo, které si myslíš, do následující množiny: ...“. Soupeř nemusí vždy odpovídat pravdivě, zalhat nám ale může pouze méně než $(m/4)$-krát. Zvítězíme, pokud poté uhodneme myšlené číslo.

   Popište vyhrávající strategii pro tuto hru.

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

Při systematickém zpracování morfologie přirozených jazyků lze postupovat různými způsoby. Nejčastější tři způsoby jsou založeny na různých základních jednotkách, jmenovitě na morfémech, lexémech nebo slovech. Vysvětlete stručně, jak každý z těchto tří způsobů funguje a pro každý z nich uveďte příklad typu jazyků, pro který je vhodný.

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

Jedním ze základních konceptů teorie Funkčního generativního popisu je valence.

1. Jaký je základní rozdíl mezi aktantem a volným doplněním?
2. Uveďte názvy alespoň 3 z 5 typů aktantů používaných v teorii Funkčního generativního popisu. Sestavte českou větu, ve které budou tyto tři typy aktantů zastoupeny.
3. Definujte valenční rámec.

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

Házíme dvěma hracími kostkami, které označujeme X a Y. Kostka X je dokonalá, tj. rozdělení hodnot z množiny $\{1, 2, 3, 4, 5, 6\}$ je rovnoměrné.

Kostka Y je falešná tak, že sudá čísla mají dvakrát větší pravděpodobnost než lichá:

$\Pr\{1\} = \Pr\{3\} = \Pr\{5\}$ a $\Pr\{2\} = \Pr\{4\} = \Pr\{6\}$, přičemž $\Pr\{2\} = 2 \cdot \Pr\{1\}$.

Čísla, která padají, interpretujeme jako hodnoty náhodných proměnných X a Y. Podle čísel na kostce Y uvažujeme také binární náhodnou proměnnou Y_b s hodnotami *sudé* a *liché*. Rozdělení náhodných proměnných je shrnuto v tabulce:

náhodná proměnná	**hodnoty**	**rozdělení**
X	$\{1, 2, 3, 4, 5, 6\}$	rovnoměrné
Y	$\{1, 2, 3, 4, 5, 6\}$	nerovnoměrné
Y_b	{*sudé*, *liché*}	nerovnoměrné

1. Co je jednotkou entropie? Co je jednotkou *podmíněné* entropie?
2. Jsou proměnné X a Y statisticky nezávislé? Svoji odpověď zdůvodněte.
3. Co můžeme říci o entropii $\mathrm{H}(Y_b)$? Vyberte jednu z následujících možností a svoji odpověď zdůvodněte.

   a) $\mathrm{H}(Y_b) > 2$ b) $\mathrm{H}(Y_b) < 2$ c) $\mathrm{H}(Y_b) = 2$

20 Rotace v prostoru (otázka studijního zaměření – 3 body)

Jde nám o metody, jak matematicky reprezentovat rotaci pevného tělesa v 3D prostoru. Středem rotace bude pro jednoduchost počátek souřadnic. Těleso se kolem počátku bude jenom otáčet, to znamená, že nebude měnit svůj tvar ani velikost ("transformace tuhého tělesa" nebo anglicky "rigid body transform").

1. Navrhněte první metodu, kterou byste zvolili pro případ, že bude třeba často rotaci interpolovat – tj. animovat dané těleso (později se budeme snažit vyjádřit plynulý a přirozený animační pohyb).
2. Navrhněte ještě jinou metodu, jak rotaci vyjádřit. Může mít proti té první některé výhody (rychlost, jednoduchost, snadnost GPU implementace), ale i nevýhody. Výhody a nevýhody uveďte a alespoň stručně zdůvodněte.
3. Jak byste postupovali při animaci rotace v čase? Vyberte si první nebo druhou metodu a dostatečně přesně popište matematický postup animace (i u této podotázky se zabývejte jen rotacemi).

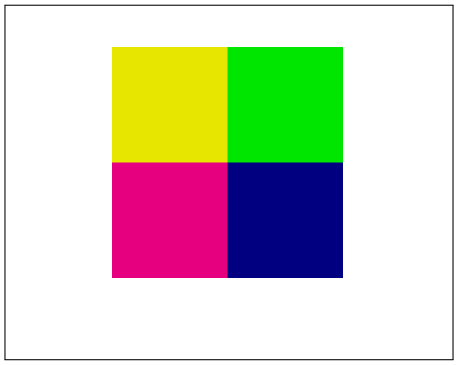
21 Stínování (otázka studijního zaměření – 3 body)

Tato otázka se týká lokálního modelu odrazu světla na povrchu tělesa v simulované 3D scéně.

1. Popište jednoduchý lokální model odrazivosti (jak barva pozorovaná na povrchu tělesa závisí na poloze a intenzitě světelného zdroje).
2. Jak se takový model odrazivosti používá v realtime zobrazení? Uvažujte běžné GPU, 3D model složený z trojúhelníků, navrhněte data posílaná ve vrcholech, jak použít GPU shadery?
3. Jak je možné dosáhnout efektu hladkého tělesa, i když je 3D model sestaven z trojúhelníků (stačí popsat myšlenku, princip)?

22 Barvy (otázka studijního zaměření – 3 body)

1. Popište modely barev RGB a HSV. Uveďte příklady použití. Popište převod RGB $\mapsto$ HSV.
2. Prohlédněte si obrázek 1. Na obrázku 2 vyznačte, která vrstva odpovídá složce R, G a B.
3. Prohlédněte si obrázek 1. Na obrázku 3 vyznačte, která vrstva odpovídá složce H, S a V.

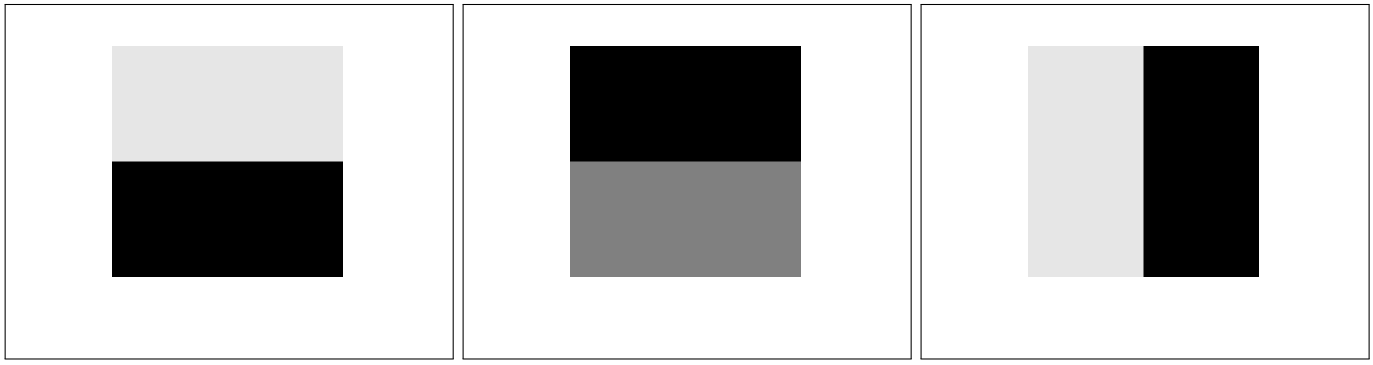


Obrázek 1: Originální obrázek.

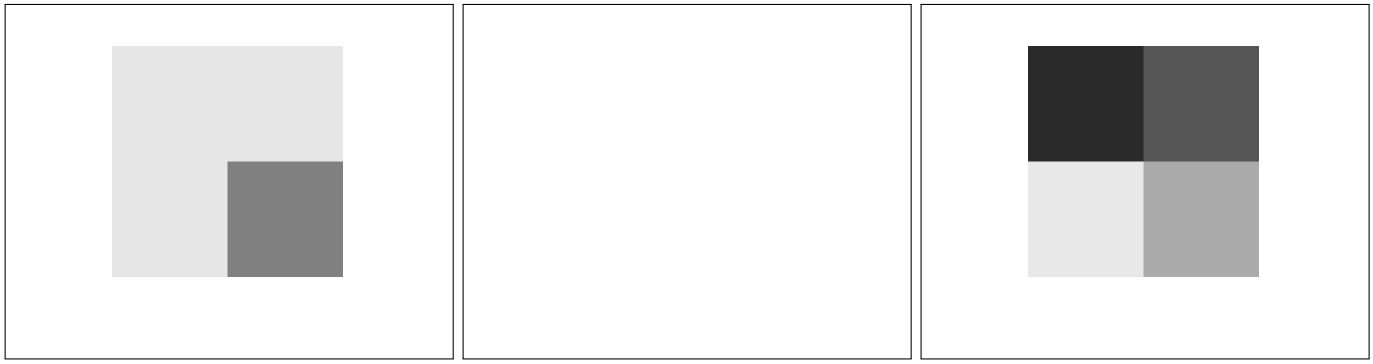
23 Webové aplikace (otázka studijního zaměření – 3 body)

Navrhněte architekturu jednoduché webové aplikace pro týmovou komunikaci typu Slack nebo WhatsApp, kterou lze využít například pro komunikaci mezi vývojáři a členy týmů v rámci softwarové firmy. Můžete si pro zjednodušení představit aplikaci, která má dvě části - server a klient. Uživatelské rozhraní klientské části zahrnuje seznam kontaktů (jména vývojářů a názvy týmů), okno ve kterém se zobrazují přijaté zprávy a další okno pro editaci nové zprávy. Každá odeslaná zpráva se hned přes server doručí všem příjemcům a zobrazí v příslušném okně klientského rozhraní. Požadovaná funkčnost také zahrnuje (1) velmi krátkou dobu odezvy na akce uživatelů a (2) možnost současného přístupu z různých zařízení (například tak, že vývojář má aplikaci spuštěnu celý den ve webovém prohlížeči na pracovním notebooku a během přestávky na oběd reaguje na urgentní zprávy přes chytrý telefon).

Zakreslete celkový návrh aplikace ve formě diagramu (grafu), kde uzly budou reprezentovat základní komponenty a hrany vyjádří interakci mezi komponentami (volání metod, posílání zpráv). Soustřeďte se na podstatné aspekty návrhu a zanedbejte



Obrázek 2: RGB prostor. Určete barevnou složku obrazu. Který obrázek odpovídá složce R, G a B? Odstíny šedé určují jakou intenzitou prispívá daná barva do výsledního barevného vjemu. Bílá - plná intenzita, černá - nulová intenzita.



Obrázek 3: HSV prostor. Určete barevnou složku obrazu. Který obrázek odpovídá složce H, S a V? Odstíny šedé určují jakou hodnotu má daná složka. Bílá - maximální hodnota, černá - nulová hodnota.

nepodstatné technické detaily. Vhodně doplňte jednotlivé prvky diagramu textovým komentářem. Dále stručně zdůvodněte architekturu a výběr použitých technologií (programovacích jazyků, knihoven, apod).

24 Návrhové vzory (otázka studijního zaměření – 3 body)

Představte si, že máte za úkol vytvořit systém pro monitorování běhu aplikací. Zásadní součástí tohoto systému je API, které budou volat samotné monitorované programy na vhodných místech (ve vhodných okamžicích) a předávat důležité informace. Uvažujte pouze objektově-orientované programovací jazyky (C#, Java, C++, atd.) na straně monitorovacího systému a také na straně aplikací, které jsou předmětem sledování. Jeden ze argumentů všech metod definovaných v API bude objekt zapouzdřující stav programu, příslušné rozhraní tohoto objektu také definuje váš systém (například `record(State st)`).

Uživatel monitorovacího systému dále bude moci konfigurovat, které informace o stavu programu se maji zaznamenávat, a to tak, že specifikuje množinu pluginů zapnutných při konkrétním běhu programu (například `plugins=memory,exceptions,...`).

Popište alespoň tři dobře známé návrhové vzory, které byste použili při vývoji takového monitorovacího systému a jeho API. Dále pro každý návrhový vzor uveďte, kde, jak a proč byste daný vzor použili (včetně ukázek v podobě fragmentů zdrojových kódů), a stručně vysvětlete princip vzoru. Ukázky použití návrhových vzorů zapište v libovolném běžném objektově-orientovaném jazyce a doplňte slovním komentářem.

25 Systémy pro správu verzí (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).

26 Garbage collection (otázka studijního zaměření – 3 body)

1. Napište algoritmus mark and sweep garbage collectoru. K popisu použijte pseudokód či přirozený jazyk podle vlastního uvážení.
2. Vysvětlete, jaké jsou vstupy algoritmu z předchozího bodu a kde se získají.
3. Na příkladu (konkrétních dat a kroků algoritmu) vysvětlete, proč se při běhu mark and sweep garbage collectoru zastavuje program, jehož heap algoritmus spravuje. K popisu opět použijte pseudokód či přirozený jazyk (jak pro popis garbage collectoru, tak pro popis příkladu programu).

27 Rozhraní pro práci se soubory (otázka studijního zaměření – 3 body)

Toto je rozhraní funkcí `mmap` a `munmap` poskytovaných operačním systémem Linux pro mapování souborů:

```
void *mmap (void *addr, size_t length, int prot, int flags, int fd, off_t offset);
int munmap (void *addr, size_t length);
```

1. Odhadněte význam argumentů a návratových hodnot funkcí. Dbejte na výstižnost odpovědi (odpovědi typu "`length` udává délku" nejsou bez dalšího vysvětlení dostačující).
2. Rozhraní vyžadují, aby hodnota argumentů `addr`, `length` a `offset` byla násobkem čísla vráceného systémovým voláním `sysconf (_SC_PAGE_SIZE)`. Vysvětlete proč.
3. Je možné mapovat celý soubor, pokud jeho velikost přesahuje velikost dostupné virtuální paměti ? Zdůvodněte.
4. Je možné mapovat celý soubor, pokud jeho velikost přesahuje velikost dostupné fyzické paměti ? Zdůvodněte.
5. Jaký je rozdíl mezi sdíleným a privátním mapováním ?

28 Specifikace vlastností programů (otázka studijního zaměření – 3 body)

1. Vysvětlete, co jsou kontrakty (preconditions a postconditions) a jaké mohou mít parametry (jaké proměnné se v nich mohou vyskytovat).
2. Vysvětlete význam invariantu cyklu v kódu níže a zdůvodněte, proč je pro verifikaci nutné tyto invarianty v kódu uvádět.
3. Pro následující kód v jazyce Pi doplňte postcondition (a pokud je třeba i precondition) tak, aby byla zachycena sémantika funkce `find`.

```
@pre  true
@post true
bool find(int[] a, int x) {
  for @ forall j.(j >= 0 && j < i -> a[j] != x) && i >= 0 && i <= |a|
      (int i := 0; i < |a|; i := i + 1) {
        if (a[i] = x)
          return true;
  }

  return false;
}
```