Bakalářské zkoušky (příklady otázek z informatiky)

podzim 2019

1 Automaty a gramatiky (3 body)

1. Uveďte dvě definice (či charakterizace), kdy je jazyk L nad abecedou Σ *regulární*.
2. Uvažme dvě následující gramatiky G_1 a G_2 nad abecedou (terminály) $\Sigma_1 = \{a, b, \neg, \vee, \wedge, (,)\}$ resp. $\Sigma_2 = \{a, b, \neg, \vee, \wedge\}$ s jediným (startovním) neterminálem S_1 resp. S_2 a pravidly

$$\begin{aligned} G_1\colon &\quad S_1 \to a \mid b \mid (\neg S_1) \mid (S_1 \vee S_1) \mid (S_1 \wedge S_1), \\ G_2\colon &\quad S_2 \to a \mid b \mid \neg S_2 \mid S_2 \vee S_2 \mid S_2 \wedge S_2. \end{aligned}$$

   Určete, zda G_1 resp. G_2 generuje regulární jazyk. Svá tvrzení zdůvodněte.

2 Prohledávání do šířky (3 body)

1. Popište datovou strukturu seznamu sousedů pro reprezentaci neorientovaného grafu.
2. Napište pseudokód algoritmu prohledávání neorientovaného grafu $G = (V, E)$ do šířky počínaje v zadaném vrcholu $s \in V$.
3. Určete časovou složitost prohledávání do šířky při použití seznamu sousedů pro reprezentaci vstupního grafu.
4. Definujte pojem bipartitního grafu a popište způsob použití průchodu do šířky pro určení toho, zda je daný neorientovaný graf bipartitní.

3 ER diagramy a dotazy v SQL (3 body)

Navrhněte (velmi) zjednodušený systém evidence letadel na letištích. Pro každé letiště chceme evidovat jeho jméno a jedinečnou zkratku. Dále chceme evidovat data a časy přistání letadel, jejich výrobce, typ a imatrikulaci (jedinečný kód letadla). Neřešte mezní situace, jako například změnu imatrikulace letadla, změnu zkratky či jména letiště, či slučování výrobců letadel.

1. Pro daný příklad navrhněte ER diagram.
2. ER diagram převeďte na relační schéma (UML diagram tříd či SQL DDL) splňující 3NF. Vyznačte klíče a cizí klíče.
3. Napište dotaz v SQL, který pro dané letiště a den vypíše žebříček výrobců letadel dle počtu přistání jejich strojů.

4 OO návrh GUI frameworku (3 body)

Vaším úkolem je v jazyce C++, C# nebo Java navrhnout základ jednoduchého okenního frameworku, který bude moci být využit k tvorbě aplikací s GUI. Vaše implementace by měla vycházet z pravidel vhodného objektového návrhu tak, aby byla do budoucna snadno rozšiřitelná a udržovatelná – nicméně, soustřeďte se na požadované vlastnosti a návrh si zbytečně nekomplikujte.

1. Váš framework by měl splňovat následující základní vlastnosti:

- Podporovat koncept okna (Window), které může obsahovat libovolné množství ovládacích prvků.
- Každý ovládací prvek je v okně umístěn na nějaké souřadnici [X, Y] a má danou šířku a výšku (tyto 4 hodnoty seskupuje předpřipravená struktura `Rectangle`, která má navíc i metodu `bool IsPointInside(Point p)`, která testuje, zda se bod `p` nachází uvnitř obdélníku).
- Připravte ovládací prvky tlačítko (Button) a uživatelem editovatelné textové pole (TextBox). Oba tyto prvky mají mít vlastnost `Text`, která reprezentuje text v nich zobrazený (jakým způsobem probíhá samotné zobrazení neřešte).

2. Každý ovládací prvek má metodu `void HandleMouseClick(Point p)`, která má být zavolána, pokud nad prvkem došlo ke kliknutí myši (metoda bude obsahovat logiku chování specifickou pro daný typ prvku; samotnou logiku ale neimplementujte; také pro jednoduchost předpokládejte, že se ovládací prvky nepřekrývají). Pro získání informace od operačního systému o tom, zda a na jakých souřadnicích došlo ke kliknutí myši, máte k dispozici hotovou třídu `OS` (vrácený bod je ve stejném souřadném systému jako souřadnice určující umístnění ovládacích prvků):

```
enum UserEvent { MouseClick, /* ... */ };
static class OS {
    public static UserEvent WaitForNextEvent() { /* ... */ }
    public static Point ReadLastMouseClick() { /* ... */ }
}
```

3. Framework musí umožňovat aplikacím potřebným způsobem reagovat na stisknutí tlačítka (detekované logikou tlačítka např. v jeho metodě `HandleMouseClick`), programátoři aplikací však nesmějí zasahovat do kódu frameworku. Aplikační kód obsluhující stisk tlačítka přitom musí být schopen komunikovat i s jinými ovládacími prvky daného okna, např. přečíst text z textového pole.

   Jako ilustraci použití vašeho frameworku napište klíčové součásti aplikace, která bude mít okno s jedním tlačítkem "Smazat" a jedním TextBoxem. Pokud uživatel do TextBoxu zadá jméno souboru a stiskne tlačítko "Smazat" (klikne na něj), tak aplikace daný soubor smaže pomocí metody `void Delete(string path)` na předpřipravené třídě `File`.

5 Překlad (3 body)

Popišme si hypotetickou architekturu procesoru. Má tři speciální registry velikosti 32 bitů:
– **SP** (Stack Pointer) - ukazatel zásobníku, ukazuje na první volné místo (prázdný zásobník má SP=0)
– **PC** (Program Counter) - ukazatel následující zpracovávané instrukce
– **FP** (Frame Pointer) - ukazatel na záznam (stack frame) funkce, viz instrukce CALL pro přesné chování

Jedinou pamětí, se kterou tento procesor pracuje, je zásobník. Rozsah platných adres je [0, SP-1], jakýkoliv přístup mimo tento rozsah je chybný. K reprezentaci čísel používá procesor celočíselný znaménkový typ o velikosti 32 bitů reprezentovaný dvojkovým doplňkem.

Instrukce procesoru jsou uvedeny v následující tabulce. V popisu používáme tyto operace a operandy:
– *imm32* - celočíselná znaménková 32-bitová konstanta
– *label* - symbolické návěští do kódu
– *PUSH(X)* - uložení hodnoty X na zásobník
– *POP(N)* - odebrání hodnoty ze zásobníku, tato operace se provádí jako N-tá v pořadí
– * - dereference odkazu

instrukce	popis	operace
ADDI4	sečte dvě celá čísla z vrcholu zásobníku a výsledek uloží zpět na zásobník	PUSH(POP(1)+POP(2))
ADDSP imm32	přičte znaménkovou konstantu k registru SP	SP=SP+imm32
SHL imm32	posune vlevo o imm32 bitů 32-bitovou hodnotu z vrcholu zásobníku	PUSH(POP(1) SHL imm32)
LLD4 imm32	uloží načtenou 32-bitovou hodnotu ze záznamu funkce na vrchol zásobníku	PUSH(*(FP+imm32))
LST4 imm32	uloží 32-bitovou hodnotu z vrcholu zásobníku do záznamu funkce	*(FP+imm32)=POP(1)
JMP label	nepodmíněný skok	PC=label
JLEI4 label	podmíněný skok, který se provede, pokud je vrchní hodnota zásobníku menší nebo rovna než hodnota pod ní	if POP(1)<=POP(2) then PC=label
LIC4 imm32	uloží 32-bitovou znaménkovou konstantu na zásobník	PUSH(imm32)
RET	návrat z funkce	FP=POP(1), PC=POP(2)
CALL label	volání funkce	PUSH(PC), PUSH(FP), FP=SP
XLD4	uloží na vrchol zásobníku 32-bitovou hodnotu, na kterou ukazuje vrchol zásobníku	PUSH(*(POP(1)))
XST4	uloží 32-bitovou hodnotu ze zásobníku na odkaz, na který ukazuje vrchol zásobníku	*(POP(1))=POP(2)

Uvažujme dále překladač z jazyka Pascal. Typ INTEGER odpovídá 32-bitovému celočíselnému typu. Klíčové slovo VAR znamená předání parametru odkazem. Předání pole odkazem je řešeno předáním odkazu na první položku pole. Jako příklad vezměme následující proceduru:

```
PROCEDURE incvar(VAR a:INTEGER);
BEGIN
  a := a + 1;
END;
```

Tu by překladač mohl přeložit například takto:

```
LLD4    -12     ; read A reference
XLD4            ; read A
LIC4    1       ; const 1
ADDI4           ; A + 1
LLD4    -12     ; read A reference
XST4            ; store A + 1
RET
```

1. Přeložte následující proceduru v Pascalu do instrukcí našeho CPU.

```
PROCEDURE addvar(a:INTEGER; b:INTEGER; VAR c:INTEGER);
BEGIN
  c := a + b;
END;
```

2. Přeložte následující proceduru v Pascalu do instrukcí našeho CPU.

```
PROCEDURE incitem5(VAR a:ARRAY[0..9] OF INTEGER);
VAR index:INTEGER;
BEGIN
  index := 5;
  a[index] := a[index] + 1;
END;
```

3. Přeložte následující proceduru v Pascalu do instrukcí našeho CPU.

```
PROCEDURE incarr(VAR a:ARRAY[0..9] OF INTEGER);
VAR i:INTEGER;
BEGIN
  FOR i := 0 TO 9 DO
    a[i] := a[i]+1;
END;
```

6 Optimalizace (3 body)

Poznámka: Bez dalšího vysvětlování můžete ve svých odpovědích použít následující větu (neformálně): Každý tok je možné dekomponovat na toky po cestách a cyklech.

1. Nechť $G = (V, E)$ je orientovaný graf a $s, t \in V$ jsou jeho dva různé vrcholy. Formulujte problém nalezení nejkratší cesty v grafu G z vrcholu s do vrcholu t jako úlohu celočíselného lineárního programování tak, že pro každou hranu $e \in E$ použijte binární proměnnou x_e a žádné další proměnné nebudete používat.
2. Uvažte lineární relaxaci Vašeho lineárního programu z bodu 1 a porovnejte hodnoty optimálních řešení původního celočíselného LP a jeho relaxace. Co o nich platí? Vysvětlete.
3. Formulujte duální program k lineárnímu programu z bodu 2; pokuste se duální program formulovat tak, aby nepoužíval více než $|V|$ proměnných.

7 Směrování (3 body)

1. Načrtněte typickou strukturu statické routovací tabulky pro protokol IPv4 a napište algoritmus, podle kterého uzel v IPv4 síti používá tuto tabulku při odesílání datagramu.
2. Vysvětlete, čím by routery omezovalo použití statické routovací tabulky a proč se používají distribuované routovací algoritmy.
3. Popište routovací algoritmus "distance vector" (používaný například v protokolu RIP).

8 Aproximační algoritmy (otázka studijního zaměření – 3 body)

1. Zaveďte pojem "aproximační faktor algoritmu".
2. Popište co nejlepší aproximační algoritmus nevyužívající lineární programování pro problém splnitelnosti a uveďte jeho aproximační poměr.
3. Dokažte uvedený odhad na aproximační faktor algoritmu z předešlé otázky.

9 Voroného diagram (otázka studijního zaměření – 3 body)

1. Definujte Voroného diagram konečné bodové množiny P v $\mathbb{R}^d$.
2. Pro každé $n \in \mathbb{N}$ a každou n-bodovou množinu P v $\mathbb{R}^2$ rozhodněte, zda má Voroného diagram množiny P alespoň jeden omezený region.

10 Kódy (otázka studijního zaměření – 3 body)

1. Definujte pojem kombinatorické (Hammingovy) koule a uveďte vzorec pro její mohutnost.
2. Formulujte Hammingův odhad velikosti pro binární kódy. Tvrzení dokažte.

11 Hypotéza kontinua (otázka studijního zaměření – 3 body)

1. Zformulujte Hypotézu kontinua. Odpovědi na následující otázky nemusíte dokazovat, stačí ANO/NE.
   (a) Je možné ji dokázat v teorii ZF?
   (b) Je možné ji vyvrátit v teorii ZF?
   (c) Je možné ji dokázat v teorii ZFC (tj. ZF s axiomem výběru)?
   (d) Je možné ji vyvrátit v teorii ZFC?

2. Napište definici spočetné množiny. Odpovědi na následující otázky nemusíte dokazovat, stačí ANO/NE. Uvažme tvrzení „sjednocení spočetně mnoha spočetných množin je vždy spočetná množina“.
   (a) Lze toto tvrzení dokázat v ZF?
   (b) Lze toto tvrzení vyvrátit v ZF?
   (c) Lze toto tvrzení dokázat v ZFC?
   (d) Lze toto tvrzení vyvrátit v ZFC?

12 Turánova věta (otázka studijního zaměření – 3 body)

1. Zformulujte Turánovu větu.
2. Nechť $t_2(n)$ je největší možný počet hran n-vrcholového grafu bez trojúhelníků. Nechť G je n-vrcholový graf bez trojúhelníků takový, že přidání libovolné hrany do G vytvoří trojúhelník. Je pravda, že G má nutně právě $t_2(n)$ hran?
3. S použitím Turánovy věty dokažte, že každý n-vrcholový graf průměrného stupně t obsahuje nezávislou množinu velikosti alespoň $n/(t+1)$.

13 Kódy (otázka studijního zaměření – 3 body)

1. Definujte pojem kombinatorické (Hammingovy) koule a uveďte vzorec pro její mohutnost.
2. Formulujte Hammingův odhad velikosti pro binární kódy. Tvrzení dokažte.

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

1. Uveďte alespoň 4 základní požadavky, které by měl splňovat dobrý program na kontrolu překlepů, a zdůvodněte je.
2. Uveďte dvě nejčastěji používané metody (pro kontrolu překlepů) založené na využití slovníku daného jazyka a vysvětlete, pro jaké typy jazyků se hodí a proč.
3. Nedílnou součástí kontroly překlepů je také nabízení vhodných oprav. Uveďte alespoň dvě prakticky použitelné metody, pomocí kterých se uživatelům budou nabízet vhodné opravy.

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

1. Unifikační gramatiky využívají speciální datový typ, tzv. Sestavu rysů (feature structure). Uveďte jeho základní vlastnosti.
2. Vysvětlete, jak funguje mechanismus unifikace dvou sestav rysů. Zdůvodněte, proč je při tvorbě unifikační gramatiky přirozených jazyků nutné používat typované sestavy rysů.
3. Nejvíce rozšířenou unifikační gramatikou byla Head Driven Phrase Structure Grammar (HPSG). Uveďte základní vlastnosti tohoto typu gramatiky.

16 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 rovnoměrné 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. Jsou proměnné X a Y statisticky nezávislé? Zdůvodněte.
2. Která z proměnných X, Y, Z má největší entropii? Odpověď přesně zdůvodněte.
3. Určete vzájemnou informaci $\mathrm{I}(X; Y)$. Výsledek zdůvodněte.

17 Scanline vyplňování (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 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).

19 2D diskrétní konvoluce (otázka studijního zaměření – 3 body)

1. Definujte (napište vzorec a popište veličiny) 2D diskrétní konvoluci.
2. Jakou konvoluční masku 3×3 byste použili pro potlačení šumu v obraze a jakou pro detekci hran? Uveďte alespoň dvě různé z každého typu.
3. Popište možné ošetření okrajového jevu.

20 IPv4 adresy (otázka studijního zaměření – 3 body)

1. Vysvětlete základní principy adresování na síťové vrstvě v TCP/IP. Popište tvar, vnitřní strukturu a zápis IPv4 adres, jejich jednotlivé třídy a způsob jejich rozpoznání. Jaké speciální adresy máme k dispozici a jaký je jejich význam?
2. Jakým (administrativním) postupem jsou přidělovány veřejné IPv4 adresy na celosvětové až lokální úrovni?
3. Jakými mechanismy se řešil či řeší nedostatek IPv4 adres?

21 Fragmentace (otázka studijního zaměření – 3 body)

1. Detailně vysvětlete problém fragmentace v protokolu IPv4 (mezi síťovou vrstvou a vrstvou síťového rozhraní) a proč k němu dochází. Jaké uzly mohou fragmentovat a jaké defragmentovat? Může být již fragmentovaný datagram znovu (ještě více) fragmentován?
2. Jakým způsobem bude fragmentován IPv4 datagram (jeho hlavička a náklad) a které položky hlavičky se k tomuto účelu použijí (nejde o jejich názvy nebo přesnou specifikaci, ale o význam a způsob využití)?
3. Jak na straně odesílatele zjistit vhodnou velikost MTU?

22 Spolehlivost (otázka studijního zaměření – 3 body)

1. Jaké dva hlavní typy chyb řešíme při zajištění spolehlivosti síťových přenosů (jeden z nich nám pomáhají řešit mechanismy jako kontrolní součet nebo CRC)?
2. Na jakém společném principu fungují mechanismy jako kontrolní součet nebo CRC?
3. Detailně vysvětlete fungování tří potvrzovacích strategií: jednotlivé potvrzování (Stop & Wait ARQ), kontinuální s návratem (Go-Back-N ARQ) a kontinuální se selektivním opakováním (Selective Repeat ARQ). V čem se tyto strategie liší a jaké mají výhody a nevýhody?

23 XML (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. 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, datum narození, kraj, ve kterém byl zvolen, a politická strana, do které patří. Pro politické strany jsou evidovány jejich názvy.

Vytvořte XML schéma (XML Schema nebo DTD) 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 poslance zvolené v Libereckém kraji.

24 Návrhové vzory (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 buď odstraňuje určitou souvislou část textu návrhu zákona nebo do stávajícího textu návrhu zákona na konkrétní jedno místo vkládá nový text nebo nahrazuje určitou souvislou část textu návrhu zákona novým textem. Podané pozměňovací návrhy jsou seřazeny do seznamu. Poslanci hlasují o jednotlivých pozměňovacích návrzích tak, jak jsou uvedeny v seznamu. Pro jednoduchost uvažujte, že dva různé pozměňovací návrhy se týkají odlišných a nepřekrývajících se částí textu a že seznam s pozměňovacími návrhy není možné nijak měnit.

Vznikl požadavek, aby měli poslanci k dispozici obrazovky, na nichž je 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.

25 Testování funkčnosti (otázka studijního zaměření – 3 body)

Stručně popište běžně využívané postupy na testování komponent (tříd, modulů), které mají netriviální závislosti. Proveďte jejich vzájemné srovnání (diskuzi výhod a omezení) z pohledu kritérií jako například vyjadřovací síla nebo jednoduchost použití během vytváření testů.

26 Rozhraní pro synchronizaci (otázka studijního zaměření – 3 body)

Následující kód představuje (z praktického hlediska naivní, ale jinak funkční) implementaci atomického čítače v C:

```
#include <pthread.h>

typedef struct {
    pthread_mutex_t mutex;
    int value;
} atomic_counter_t;

void counter_init (atomic_counter_t *counter) {
    pthread_mutex_init (&(counter->mutex), NULL);
    counter->value = 0;
}

int counter_fetch_and_increment (atomic_counter_t *counter) {
    pthread_mutex_lock (&(counter->mutex));
    int snapshot = counter->value ++;
    pthread_mutex_unlock (&(counter->mutex));
    return (snapshot);
}
```

1. Upravte původní implementaci tak, aby místo mutexu (reprezentovaného typem `pthread_mutex_t`) používala semafor (reprezentovaný typem `sem_t`). Pro připomenutí, zde jsou vybrané signatury funkcí pro práci se `sem_t`:

   ```
   #include <semaphore.h>

   int sem_close (sem_t *sem);
   int sem_destroy (sem_t *sem);
   int sem_getvalue (sem_t *sem, int *sval);
   int sem_init (sem_t *sem, int pshared, unsigned int value);
   sem_t *sem_open (const char *name, int oflag);
   sem_t *sem_open (const char *name, int oflag, mode_t mode, unsigned int value);
   int sem_post (sem_t *sem);
   int sem_trywait (sem_t *sem);
   int sem_unlink (const char *name);
   int sem_wait (sem_t *sem);
   ```

2. S použitím podmínkových proměnných (reprezentovaných typem `pthread_cond_t`) rozšiřte původní implementaci o funkci
   `void counter_wait_for_watermark (atomic_counter_t *counter, int watermark)`, která se vrátí ve chvíli, kdy hodnota čítače dosáhne `watermark`. Pro připomenutí, zde jsou vybrané signatury funkcí pro práci s `pthread_cond_t`:

   ```
   #include <pthread.h>

   int pthread_cond_broadcast (pthread_cond_t *cond);
   int pthread_cond_destroy (pthread_cond_t *cond);
   int pthread_cond_init (pthread_cond_t *restrict cond, const pthread_condattr_t *restrict attr);
   int pthread_cond_signal (pthread_cond_t *cond);
   int pthread_cond_wait (pthread_cond_t *restrict cond, pthread_mutex_t *restrict mutex);
   ```

3. Typ `atomic_counter_t` byl navržen pro použití vlákny v rámci jednoho procesu. Jak byste jej museli upravit či rozšířit, aby bylo možné jej použít vlákny různých procesů (stačí uvést potřebné kroky, není nutné psát implementaci) ?

27 Garbage collection (otázka studijního zaměření – 3 body)

1. Formulujte generační hypotézu a vysvětlete, proč je důležitá pro optimalizaci garbage collection algoritmů.
2. Následující třída implementuje jednoduchý servlet objekt v jazyce Java. Jeho rolí je obsluhovat příchozí požadavky na webovém serveru (funkci lze intuitivně odhadnout, při startu serveru se vytvoří instance servletu spojená s konkrétním URL, tato instance je pak volána pro obsluhu jednotlivých HTTP požadavků, argumenty reprezentují přenášená data).

```
import java.util.Map;
import java.util.HashMap;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MyServlet extends HttpServlet
{
    private Map<String, Integer> stats = new HashMap<String, Integer> ();

    @Override public void doGet (HttpServletRequest request, HttpServletResponse response)
    throws java.io.IOException
    {
        int count;
        String query = request.getQueryString ();
        synchronized (this) {
            count = stats.getOrDefault (query, 0) + 1;
            stats.put (query, count);
        }

        response.setContentType ("text/plain");
        java.io.PrintWriter output = response.getWriter ();
        output.println ("Query␣" + query + "␣number␣" + count + ".");
    }
}
```

   U každé reference v uvedené třídě rozhodněte, zda pravděpodobně ukazuje do mladé či staré generace a vysvětlete proč.

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

Následující kód obsahuje část rozhraní operačního systému Linux pro práci se soubory:

```
int open (const char *pathname, int flags, mode_t mode);
int close (int fd);
ssize_t read (int fd, void *buf, size_t count);
ssize_t readv (int fd, const struct iovec *iov, int iovcnt);
ssize_t write (int fd, const void *buf, size_t count);
ssize_t writev (int fd, const struct iovec *iov, int iovcnt);

struct iovec {
    void  *iov_base;    /* Starting address */
    size_t iov_len;     /* Number of bytes to transfer */
};
```

1. Vysvětlete rozdíl mezi funkcemi `read` a `readv`, respektive `write` a `writev`. Proč tyto dvě varianty funkcí existují a v jakých situacích by se typicky použily ?
2. Je možné volání funkce `readv` nahradit voláním(i) funkce `read`, respektive volání funkce `writev` voláním(i) funkce `write` ? Vysvětlete proč.
3. Může se stát, že kvůli omezením hardware nebudou funkce `readv` a `writev` dostupné ? Vysvětlete proč.