Kód studenta 69

počet listů odpovědi (pokud více než tento jeden)

1 Algoritmy rozděl a panuj (společné okruhy)

1) Mějme algoritmus A, který na datech velikosti n udělá $T(n)$ elementárních operací. Algoritmus A je rekurzivní a pracuje takto (a, c, x, y jsou přirozená čísla, $a > 0$, $c > 1$):

a) Udělá $\Theta(n^x)$ elementárních operací, aby ze vstupních dat vybral a podmnožin velikosti n/c.

b) Rekurzivně pustí sám sebe na každou z vybraných podmnožin dat (pokud má velikost alespoň c, pro data menší velikosti vyřeší úlohu v konstantním čase).

c) Udělá $\Theta(n^y)$ elementárních operací, aby všechna řešení z bodu b) spojil do řešení původní úlohy na datech velikosti n.

Určete (bez důkazu) asymptoticky těsný odhad funkce $T(n)$ v závislosti na parametrech a, c, x, y.

2) Nechť X a Y jsou pole délky n, každé obsahující *setříděnou* posloupnost n přirozených čísel. Navrhněte a popište algoritmus s časovou složitostí $O(\log n)$, který najde medián (jeden z mediánů) všech $2n$ čísel obsažených v polích X a Y. Dokažte metodou z bodu 1), že složitost Vašeho algoritmu je opravdu $O(\log n)$.

Kuchařková věta: $T(n) = a \cdot T\left(\frac{n}{b}\right) + \Theta(n^c)$ $a, b > 0$ $c \geq 1$

$q := \frac{a}{b^c}$

$q > 1$ $\Theta(\log_a b)^3 \cdot T(n)$ $\log_b a$

$q < 1$ $\Theta(n^c) \ni T(n)$

$q = 1$ $\Theta(n^c \log(n)) \ni T(n)$

---

1) alg. A: $T(n) = a \cdot T\left(\frac{n}{c}\right) + \Theta(n^y + n^x)$

$q = \frac{a}{c}$

$q > 1 \Rightarrow T(n) \in \Theta(\log_a c)$

$q < 1 \Rightarrow T(n) \in \Theta(n^y + n^x)$

$q = 1 \Rightarrow T(n) \in \Theta((n^y + n^x) \cdot \log(n))$

$x \leq y \Rightarrow T(n) \in \Theta(n^y)$

$x > y \Rightarrow T(n) \in \Theta(n^x)$

$x \leq y \Rightarrow T(n) \in \Theta(n^y \log(n))$

$x > y \Rightarrow T(n) \in \Theta(n^x \log(n))$

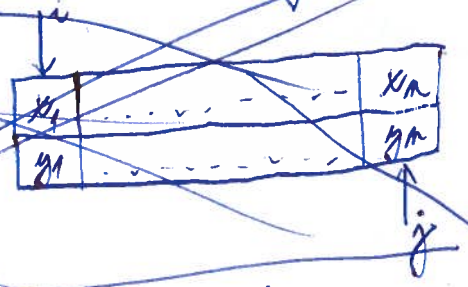
$\frac{a}{c^x} \rightarrow f(x, y)$

2) Idea:

Představme si X, Y v tabulce jako řádky

$X = (x_1, \dots, x_n)$

$Y = (y_1, \dots, y_n)$



Máme ukazatele $X = [x_1, \dots, x_n]$, $Y = [y_1, \dots, y_n]$

Idea: X, Y obsahují hodnoty o nějakém intervalu. Pokud se intervaly neprotínají, pak triviálně $x_n \leq y_1$ nebo $y_n \leq x_1$

| X

medián = x_n

medián = y_n

Uvažujme případ, kdy se intervaly protínají tak, že $x_n = y_1$ nebo $y_n = x_1$

pak triviálně medián = x_n, resp. y_n.

Kód studenta 69

2 počet listů odpovědi (pokud více než tento jeden)

2 Semafor (společné okruhy)

Knihovna `System.Threading` jazyka C# obsahuje třídu `Semaphore` s metodami odpovídajícími operacím klasického semaforu, *P* (`WaitOne`) a *V* (`Release`). Objekt vytvořený pomocí `new Semaphore(0, 1)` odpovídá binárnímu semaforu inicializovanému nulou. Pokud více než jedno vlákno čeká na tentýž semafor, pořadí, ve kterém budou vlákna spouštěna po uvolnění semaforu, není specifikováno.

Program vpravo má produkovat výstup `OneTwoThreeFourFive`, přitom liché části výstupu mají být vypisovány funkcí `f1` a sudé části funkcí `f2`, přičemž tyto funkce běží v různých vláknech.

Synchronizace, implementovaná pomocí semaforu, téměř funguje, ale obsahuje časově závislé chyby (race conditions).

1. Popište scénář, který vede k jinému výstupu než `OneTwoThreeFourFive`, nebo končí uváznutím (deadlock). Scénař zapište jako posloupnost čísel řádek, přičemž každé číslo reprezentuje *ukončení* příkazu na dané řádce. Pouze vnitřky funkcí `f1` a `f2` jsou relevantní.
2. Existovaly by v tomto kódu časově závislé chyby i v případě, že by knihovna C# garantovala FIFO pořadí spouštění vláken čekajících ve `WaitOne`?
3. Napište řešení, které neobsahuje časově závislé chyby a spolehlivě vypisuje požadovaný výstup `OneTwoThreeFourFive`. Použijte přitom dva semafory a pouze jejich funkce `Release()` and `WaitOne()`. Řešení nesmí používat žádné jiné proměnné nebo objekty sdílené mezi vlákny kromě těchto dvou semaforů. Volání `Console.Write` musejí samozřejmě zůstat tam, kde jsou.

(Jazyk C# je v této otázce použitý pouze jako generický zástupce běžných programovacích jazyků, otázka ani řešení s jazykem C# jako takovým nesouvisí.)

```
1  class Program
2  {
3      private static Semaphore sem;
4
5      private static void f1()
6      {
7          Console.Write("One");
8          sem.Release();
9          sem.WaitOne();
10         Console.Write("Three");
11         sem.Release();
12         sem.WaitOne();
13         Console.Write("Five");
14     }
15
16     private static void f2()
17     {
18         sem.WaitOne();
19         Console.Write("Two");
20         sem.Release();
21         sem.WaitOne();
22         Console.Write("Four");
23         sem.Release();
24     }
25
26     static void Main(string[] args)
27     {
28         sem = new Semaphore(0, 1);
29         Thread t1 = new Thread(f1);
30         Thread t2 = new Thread(f2);
31         t1.Start();
32         t2.Start();
33         t1.Join();
34         t2.Join();
35     }
36 }
```

NELZE, sem JE INICIALIZOVÁN 0

1) (18) 19,20,21,22,23, 7,8,9,10,11,12,13 — NEVALIDNÍ SCÉNÁŘ se rozběhnou

2) Ano - když startujeme vlákna, tak pořadí, ve kterém běží, je nedeterministické
⇒ může nastat scénář začínající 7,18,19 i 18,19,20,7 TO NENÍ ODPOVĚĎ NA TUTO OTÁZKU

AUTOR PŘEDPOKLÁDÁ OBRÁCENÝ SMYSL OTÁZKY
TÍM SE PODOTÁZKY 1 A 2 STÁVAJÍ TRIVIÁLNÍ.

Kód studenta: 69
Otázka: 2 Semafor

3)

```
class Program
{
private static Semaphore oddSem;
private static semaphore evenSem;
static void Main(string[] args)
{
    oddSem = new Semaphore(1,1);
    evenSem = new semaphore(1,1);
    Thread t1 = new Thread(f1);
    Thread t2 = new Thread(f2);
    t1.Start();
    t2.Start();
    t1.Join();
    t2.Join();
}
private static void f1()
{
  Console.Write("One");
  evenSem.Release();
  oddSem.WaitOne();
  Console.Write("Three");
  evenSem.Release();
  oddSem.WaitOne();
  Console.Write("Five");

}
private static void f2()
{
  evenSem.WaitOne();
  Console.Write("Two");
  oddSem.Release();
  evenSem.WaitOne();
  Console.Write("Four");
  oddSem.Release();

}
}
```

OK (PRO OBRÁCENÉ SEMAFORY)

Kód studenta 69

☐ počet listů odpovědi (pokud více než tento jeden)

3 Báze vektorových prostorů (společné okruhy)

1. Zformulujte Steinitzovu větu o výměně vhodných vektorů mezi lineárně nezávislou a generující množinou (čili nikoli nutně bázemi).
2. Mějme reálnou matici $\boldsymbol{A}$ takovou, že je podobná diagonální matici $\boldsymbol{D}$ prostřednictvím součinu $\boldsymbol{R}^{-1}\boldsymbol{A}\boldsymbol{R}$, kde

$$\boldsymbol{R} = \begin{pmatrix} -1 & -1 & -8 & -2 & -8 \\ 0 & 1 & -3 & 0 & 2 \\ -1 & -2 & 6 & -2 & 3 \\ -1 & -3 & 7 & -1 & -4 \\ 1 & 2 & 0 & 2 & 4 \end{pmatrix},$$

přičemž je známo, že prvky na diagonále $\boldsymbol{D}$ jsou čísla $17, 17, 23, 17, 23$ v uvedeném pořadí.

Rozhodněte, zdali některý ze sloupců $\boldsymbol{R}$ lze vyměnit za následující vektor $\boldsymbol{u_i}$, aby matici $\boldsymbol{A}$ bylo stále možné diagonalizovat i prostřednictvím výsledné matice $\boldsymbol{R}'$.

Pokud taková výměna je možná, určete všechny sloupce matice $\boldsymbol{R}$, které mohou být vyměněny.

(a) Řešte pro $\boldsymbol{u_1} = (4, 2, 2, -3, -2)^{\mathrm{T}}$.

(b) Řešte pro $\boldsymbol{u_2} = (7, -6, -2, 12, -6)^{\mathrm{T}}$.

Své odpovědi zdůvodněte.

Kód studenta 69

☐ počet listů odpovědi (pokud více než tento jeden)

4 Model teorie (společné okruhy)

1. Nechť T je teorie jazyka (signatury) $L = \langle \mathcal{R}, \mathcal{F} \rangle$ v predikátové logice (kde $\mathcal{R}, \mathcal{F}$ jsou množiny relačních a funkčních symbolů s danými aritami). Uveďte definice pojmů *struktura jazyka L* a *model teorie T*.
2. Vyjádřete následující tvrzení formulemi $\varphi_1, \varphi_2, \varphi_3$ v jazyce $L = \langle Z, S, P \rangle$ predikátové logiky (s unárními predikáty pro *'složit zkoušku'*, *'mít štěstí'*, *'být připraven'*).
   (a) *Ne každý, kdo složí zkoušku, má štěstí, ale kdo má štěstí, zkoušku složí.*
   (b) *Štěstí přeje připraveným. (Kdo je připraven, má štěstí.)*
   (c) *Nějaký student byl připraven, ale zkoušku nesložil.*
3. Pro teorii $T_1 = \{\varphi_1, \varphi_2\}$, a také pro teorii $T_2 = \{\varphi_1, \varphi_2, \varphi_3\}$, buď najděte nějaký její model anebo formálně dokažte (pomocí tablo metody, rezoluce, či Hilbertovského kalkulu), že žádný model nemá.

2) a) $\varphi_1 = (S(x) \to Z(x))$

b) $\varphi_2 = P(x) \to S(x)$

c) $\varphi_3 = P(A) \wedge \neg Z(A)$

určitého reprezentujícího studenta

1) model teorie T je takové, že všechny formule teorie jsou pravdivé

3) model T_1: $Z(x) = 1$ $S(x) = 1$ $P(x) = 1$

3) T2 nemá model:

$$T\,(S(x) \to Z(x)) \wedge (P(x) \to S(x)) \wedge (P(a) \wedge \neg Z(a))$$

v tabl. musí být sentence

- $T\ S(x) \to Z(x)$
- $T\ P(x) \to S(x)$
- $T\ P(a) \wedge \neg Z(a)$
- $T\ P(a)$
- $T\ \neg Z(a)$
- $F\ Z(a)$

není konečný krok

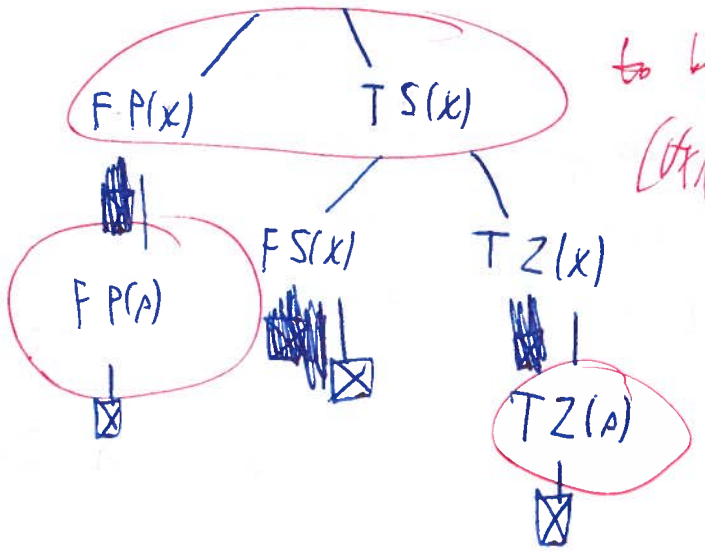
takto tablo metoda v PL nefunguje, nelze ignorovat kvantifikátory!

to by tabl. i validní konk.

$(\forall x)(P(x) \to S(x)) \wedge (\forall x) P(x)$

nebo ∨

$(\forall x) S(x)$



odkud ?

3-

Kód studenta 69

2 → počet listů odpovědi (pokud více než tento jeden)

5 Organizace paměti (specializace PVS)

Uvažujte následující program v C/C++. Předpokládejte 32-bitový procesor a velikosti datových typů char 1B, int 4B, double 8B.

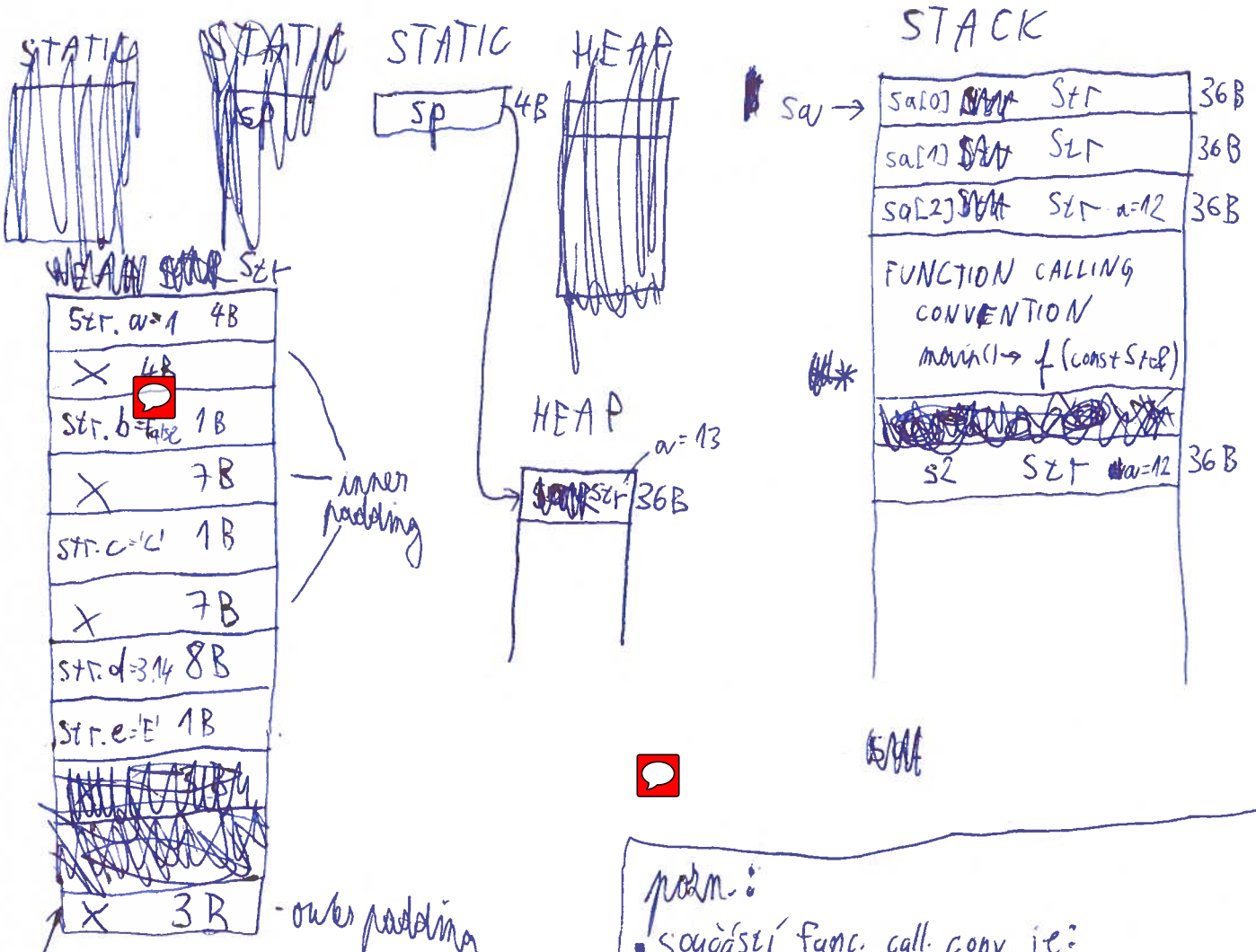
```
struct Str { int a = 11; bool b = false; char c = 'C'; double d = 3.14; char e = 'E'; };

Str * sp = 0;

int f( const Str& s1) {
        *sp = s1;
        Str s2 = *sp;
        ++(*sp).a;
        // memory dump
        return s2.a;
}

int main() {
        Str sa[3];
        sp = new Str;
        ++sa[2].a;
        sa[1].a += f( sa[2]);
        delete sp;
        sp = 0;
}
```

Nakreslete a popište s přesností na bajty rozložení paměti pro všechna data programu a její obsah (tam, kde ho znáte) při zastavení programu v místě označeném `// memory dump`. Jasně odlište různé druhy paměti za běhu programu (zásobník apod.). Tam, kde je to relevantní, stručně popište možné alternativy. Schematicky znázorněte i interní data nutná při běhu programu, i když neznáte jejich přesnou strukturu ani hodnoty. Pokud budete pro zákres a popis potřebovat konkrétní adresy, vhodně si je vymyslete. Adresu na konkrétní znázorněné paměťové místo lze zakreslit šipkou. Přesnou reprezentaci hodnot typu double řešit nemusíte.

obecná org. paměti:

STATIC - glob. proměnné
HEAP - dynamicky alokovaná paměť
STACK - ostatní - př. lok. prom., call stack, ...
STATIC - fixní vel.
STACK, HEAP - dle potřeby za běhu zvětšíme/zmenšíme využívanou paměť, až do vyhrazeného limitu

Kód studenta: 69
Otázka: 5 Org. paměti



Takto vypadá struktura Str v paměti s default hodnotami (inner padding zarovnává na násobky velikosti největšího fieldu, outer padding za posledním fieldem zarovnává paměť na násobky 4)

vel. STR = 36B

pozn.:

- součástí func. call. conv. je:
  - místo v kódu, kam se vrátíme po skončení volané fce - 4B
  - parametry volané fce

tedy na našem zás. je v bloku označeném * parametr s1 o vel. 4B (ref.) ukazující do sa[2]

- sa[2].a má hodnotu 12
  - hodn. zvětšujeme v main()
  - ve fci f nejprve do Str na heapu kopírujeme obsah místa, kam ukazuje ref. s1
  - následně ve fci f modifikujeme Str, který se nachází v HEAP

=> sp→a je rovno 13

Kód studenta 69

3 b

2 počet listů odpovědi (pokud více než tento jeden)

6 Paralelní výpočty a synchronizace vláken (specializace PVS)

Popište tyto prostředky na synchronizaci výpočtů (akcí) několika současně běžících vláken: atomické operace, atomické proměnné. Vysvětlete jejich sémantiku především s ohledem na možné stavy programu a uveďte příklad použití v jednom z programovacích jazyků C++, C#, Java.

Popište tyto abstrakce pro vyjádření paralelních výpočtů: task, fork-join a future. Vysvětlete jejich sémantiku a uveďte příklad použití v jednom z programovacích jazyků C++, C#, Java.

Hodnotí se především správné vyjádření konceptů, drobné syntaktické nepřesnosti v příkladech nejsou podstatné.

atomické operace - proběhnou celé nebo neproběhnou vůbec

~~jakmile začnou, nelze je přerušit~~

- máme jistotu, že prostředky, se kterými operace pracuje, nebudou dostupné jiným vláknům než tomu, které tuto operaci provádí

- např. atomický inkrement - máme jistotu, že jiné vlákno nebude číst ani psát do dat, dokud inkrement nebude zcela hotový

atomické proměnné - proměnné, na kterých lze provádět jen atomické operace

- např. atomický int - jeho inkrementace je implicitně atomická

(inkrement se skládá z načtení dat, úpravy dat a zápisu dat - ~~atomický provede vše naráz~~ u atomického je představa t.ž. provede vše naráz)

klasický příklad použití - suma hodnot pole

---

př.

```
class Program
private static int counter;
private static AtomicInt atomicCounter;
private static object counterLock = new();
static void Main(string[] args)
{
  Thread t1 = new Thread(AtomicIncr);
  Thread t2 = new Thread(AtomicIncr);
  Thread t3 = new Thread(Incr);
  Thread t4 = new Thread(Incr);
t1.Start(); t2.Start(); t3.Start(); t4.Start();
```

paraleln, sum a hodnot pole: (prikaz s atom. oper. a atom. prom.)

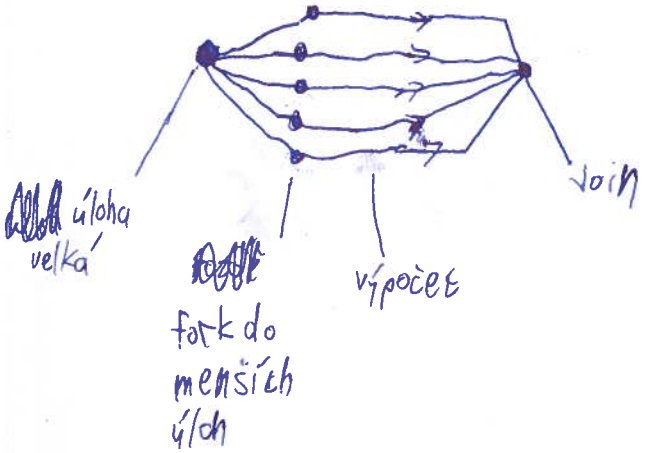
```
class Program
{
  static int sum;
  static object sumLock = new();
  static AtomicInt atomicSum;
  static void Sum(int[] a, int from, int to)
  {
    for(int i=from; i<to; ++i)
    {
      lock(sumLock)  // Atomic value update
      {
        sum += a[i];
      }
    }
  }
  static void AtomicVarSum(int[] a, int from, int to)
  {
    for(int i=from; i<to; ++i)
    {
      atomicSum.Add(a[i]);
    }
  }
  static void Main(string[] args)
  {
    int[] a = new int[] {0,1,2,3,4};          a.Count/2
    Thread t1 = Thread.Run(()=>{Sum(a,0,a.Count/2);});
    Thread t2 = Thread.Run(()=>{Sum(a,a.Count/2,a.Count);});
    Thread t3 = Thread.Run(()=>{AtomicVarSum(a,0,a.Count/2);});
    Thread t4 = Thread.Run(()=>{AtomicVarSum(a,a.Count/2,a.Count);});
    t1.Join(); t2.Join(); t3.Join(); t4.Join();
  }
}
```

Kód studenta: 69

Otázka: 6 Para výpočty a synch. vláken

Task = úloha, která má běžet na nějakém vlákně

Fork-join = máme úlohu, jejíž výpočet rozběhneme na několika vláknech (to je Fork)
poté čekáme, až vlákna doběhnou (všechna), a zkontrolujeme výsledek (to je Join)



Future = slib, že někdy v budoucnu dostaneme něco (typicky výsledek výpočtu-hodnotu)
- se slibem můžeme pracovat, jako by tam ta hodnota již byla (např. ho posílat dále)
- jakmile ale chceme tu hodnotu přesně, musíme si počkat

---

př.:

```
AtomicInt sum;
int[] a = new int[] {0,1,2,3,4,5,10,11};
Parallel.For(0, a.Count, i => {
    sum.Add(a[i]);
});
```

Parallel.For rozběhne vlákna, přičemž:
Parallel.For představuje Fork-Join v C#:
rozběhne vlákna a čeká, než doběhnou.

př. Task:

```
var t1 = Thread.Run(() => {/*computation*/});
```

↑ reprezentace Tasku v C#

př. Future:

```
Thread<int> FutureSum(int[] arr)
{
    var tcr = new TaskCompletionSource(() =>
    {
        int sum = 0
        foreach(var value in arr)
        {
            sum += value;
        }
        return sum
    });
    return tcr.Task;
}

int[] a = new int[] {0,1,2};
var futureSum = FutureSum(a); // získáme slib

...

var sum = futureSum.Value; // čekáme na dodržení slibu
```

Kód studenta 69

1

2 počet listů odpovědi (pokud více než tento jeden)

7 SQL (specializace PVS)

Předpokládejme následující relační schéma **databáze zoologické zahrady**:

Zamestnanec (rodneCislo, jmeno, prijmeni, oddeleni)
Primární klíč: rodneCislo
Zvire (jmeno, druh, vek, zemePuvodu)
Primární klíč: jmeno, druh
Osetrovani (id, jmeno, druh, zamestnanec, datumZacatku, datumKonce)
Primární klíč: id
Unikátní klíč: jmeno, druh, zamestnanec, datumZacatku
Cizí klíč: Osetrovani[zamestnanec] ⊆ Zamestnanec[rodneCislo]
Cizí klíč: Osetrovani[jmeno, druh] ⊆ Zvire[jmeno, druh]

Vytvořte SQL SELECT výrazy pro následující dotazy:

1. Najděte unikátní rodná čísla zaměstnanců, kteří alespoň v jednom okamžiku ošetřovali alespoň dvě zvířata najednou.
2. Najděte křestní jména a příjmení zaměstnanců z oddělení *kočkovitých šelem*, kteří nikdy neošetřovali nějaké zvíře mající zemi původu *Maroko* nebo *Tunisko*.

~~1. SELECT~~

1. SELECT DISTINCT rodneCislo
FROM Osetrovani JOIN Zamestnanec ON Osetrovani.Zamestnanec = Zamestnanec.rodneCislo
~~WHE~~

1. SELECT DISTINCT ~~AVG~~ zamestnanec
FROM Osetrovani o1 JOIN Osetrovani o2 ON o1.Zamestnanec = o2.zamestnanec
WHERE (o1.jmeno != o2.jmeno ~~AND~~ OR o1.druh != o2.druh)
AND o1.datumZacatku ≤ o2.datumKonce ... ✗

~~2. SELECT za.jmeno~~

KÓD STUDENTA: 69
ČÍSLO OTÁZKY: 7 SQL

2.

```
SELECT DISTINCT za.jmeno, za.prijmeni
FROM Osetrovani o JOIN Zamestnanec za ON(o.zamestnanec = za.rodneCislo
     JOIN Zvire zv ON(o.jmeno = zv.jmeno AND o.druh = zv.druh)
WHERE za.oddeleni = "Kočkovité šelmy"
     AND za
```

TAKTO TO NEPŮJDE

1+

Kód studenta 69

☐ počet listů odpovědi (pokud více než tento jeden)

8 Analýza požadavků a návrh uživatelských testů (specializace PVS)

Pracujete na nové platformě pro streamování filmů. Ze schůzky se stakeholdery jste si přinesli následující poznámku: "Naši uživatelé chtějí mít možnost jednoduše najít filmy, které je zajímají. Někteří uživatelé chtějí vyhledávat klasicky podle názvu, roku vydání, žánru, apod. Toto klasické vyhledávání tedy do platformy zavedeme. Je ale vhodné jen pro uživatele, kteří vědí, co hledají. Pro řadu uživatelů je to složité, protože přesně nevědí, na co by se chtěli podívat. Z průzkumů jsme zjistili, že by uživatelé také chtěli, aby jim naše platforma doporučovala filmy podle streamovací historie jejich přátel. Zavedeme tedy pro každého uživatele možnost si zobrazit seznam takových doporučení. K tomu musíme mít od uživatele povolení, zda si můžeme načíst seznamy jeho přátel z jeho sociálních sítí. Ze začátku chceme umět jen načítání ze sociálních sítí podporujících Facebook Graph API, ale to musí být snadno rozšiřitelné i na další druhy API."

1 1. Zahrnuje poznámka pouze funkční požadavky? Odpovězte ano/ne a poté vysvětelete.

0 2. Identifikujte alespoň 2 různé funkční požadavky a vyjádřete je jako user stories.

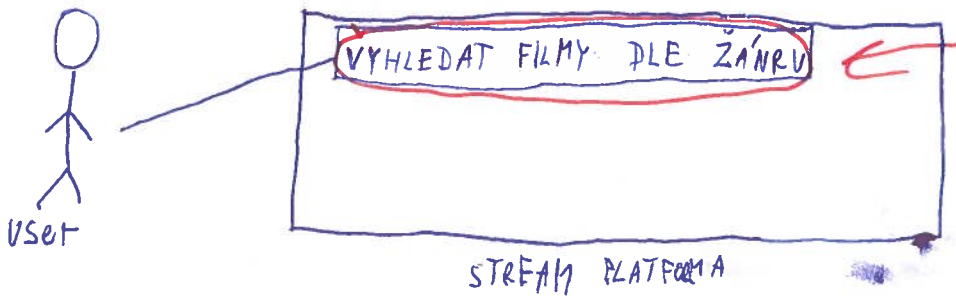
3. Vyberte si jeden z vašich user stories a vyjádřete jej jako strukturovaný use case (případ užití). Uveďte všechny části popisu use case, které typicky uvažujeme.

0+ 4. Pro váš use case navrhněte test case, který použijeme k otestování implementace tohoto use case.

1. NE - ~~na konci~~ na konci poznámky píšeme, že chceme ~~mít~~ lehkou rozšiřitelnost na další druhy API, což je kvalitativní požadavek.

2. ~~Jako uživatel chci mít možnost jednoduše najít filmy, které mě z~~
Jako uživatel chci mít možnost vyhledávat filmy podle žánru, abych mohl najít filmy, které mě zajímají.
Jako uživatel chci mít možnost zobrazit doporučené filmy dle streamovací historie mých přátel, ~~abych se při výběru filmu mohl inspirovat~~, pro případ, že nevím přesně, na co se chci dívat.

3.



use case diagram, ale ten jsme nechtěli

4. PRE-CONDITION:
Uživatel má otevřené ~~vyhledávací formulář~~ vyhledávání filmů

TEST: 1. Uživatel zadá žánr filmů, na které se chce podívat.
2. Uživateli se zobrazí seznam filmů s daným žánrem.

co testuje - z (test cond.)
chybí rozpad do atomických kroků a jejich ověření (ověření filt. obr. + vid. ...)