



(3) MM



## Kód studenta 46

### 1 Třídění sléváním (3 body)

- Formulujte algoritmus třídění posloupnosti  $n$  celých čísel sléváním (Mergesort) a zapište ho v pseudokódu.
- Analyzujte časovou a prostorovou složitost tohoto algoritmu.
- Navrhněte efektivní algoritmus pro slévání  $k$  setříděných posloupností o celkem  $n$  prvcích. Upravte Mergesort, aby toto slévání používal. Jak se změní jeho časová složitost? Zapište ji jako funkci proměnných  $n$  a  $k$ .

```

1] int[] MergeSort (int[] arr) {
    int length = arr.length;
    if (length > 1) {
        int mid = length / 2 - 1;
        int[] low = MergeSort (arr[0:mid]);
        int[] high = MergeSort (arr[mid+1:length]);
        return Merge (low, high);
    }
    return arr;
}

int[] Merge (int[] low, int[] high) {
    int lowLen = low.length;
    int highLen = high.length;
    int lowIndex = 0, highIndex = 0, mergedIndex = 0;
    int[] merged = new int [lowLen + highLen];
    while (lowIndex < lowLen && highIndex < highLen) {
        if (low[lowIndex] < high[highIndex])
            merged[mergedIndex] = low[lowIndex];
        lowIndex++;
    } else {
        merged[mergedIndex] = high[highIndex];
        highIndex++;
    }
    mergedIndex++;
    while (lowIndex < lowLen) merged[mergedIndex++] = low[lowIndex++];
    while (highIndex < highLen) merged[mergedIndex++] = high[highIndex++];
    return merged;
}

```

Nevyrene pole,  
rozdelime na 2  
jedny a na ně  
splijeme slijem  
funkci, vrati se nám  
sortovani usely a by  
splijeme uveden Merge  
je  $\Theta(n)$ .

parametry:  
int[] arr = ...;  
arr = MergeSort (arr);

Pohled je na  
usely binární  
-muzí relativně  
1 - vzdálené  
usely.

OK

2)

časová složitost:

je dvojnásobnou obecností  $m = 2^k$  (tedy by se pak je mnoha omezena nějakým  $2^k$ , které je největší dvojnásobek něčeho, což je konstanta)

Funkce 'Merge' běží v lineárním čase k délce vstupu. Funkce MergeSort pracuje na vstupu velikosti  $n$   $\Theta(n)$  a vstup velikosti  $n > 1$  rozděluje na 2 stejně velké části a dle následující velikosti  $M/2$ . Této se tedy napsal

$\Theta(T(1)) = 1$  a  $T(m) = 2 T(M/2) + \Theta(n)$ . Podle 'kudrálkové nějaký' je

$$\begin{array}{ccc} 2 \xrightarrow{\text{části}} & \xrightarrow{n} & \xrightarrow{n} \\ & \text{velké} & \text{funkce Merge} \\ & M/2 & \end{array}$$

$\Theta = \frac{2}{2} = 1$  a tedy  $T(n) \in \Theta(n \log n)$ . Jinak řečeno, Strom rekurrenci má složku  $\log n$ , a když každou je  $2^k$  částí a na každou potřebujeme  $\frac{M}{2^k}$  času na Merge, a každou tedy  $\Theta(n)$  času a každá koule je  $\Theta(n \log n)$ .

Nejdůležitější poznámka:

nejdůležitější poznámka  
všechny najednou

Nejdůležitější poznámka, tedy ne k tomu každou samou  $2^k$  dležití velikosti  $\frac{m}{2^k}$  tedy  $\Theta(n)$  ne každou, každou je  $\log n$ , sloučení všechna dolešší koupi, a nejdůležitější výsledné tedy  $\Theta(2^n \log n) = \Theta(n \log n)$ .

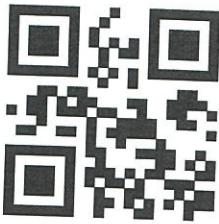
Poznámka: tedy  $\Theta(n \log n)$  podobně. Dáme 2 koupe velikosti  $n$ , tedy písací/přijímací do druhého a slévání v prvním (slévání koupi). **OK**

3) Zde mám stáčí funkce Merge, když máme výsledky 2 sloučené počítat a dostavovat sloučené počítat. Funkce MergeSort pak ve svém posloupství málo je do fronty, upředu 2, navráte se mi funkce Merge a následně vložit do fronty. Pakd ve frontě je jen jeden front, máme sloučené.

Nejdříve jde o předcházející výsledků, nechtěj je BÍNO  $k=2^l$ . Po slítí ke koupi máme  $k/2$  koupe velikosti  $2^k$ , vložení 1 vše velikosti  $2^{\log_2 k} = k$ . Po slítí koupi frontu máme tedy  $2^k$  koupe menší koupi tedy složitost je  $\Theta(k \cdot \log_2 k)$ , ne,  $n$  je celková délka pro  $\Theta(k \cdot \log_2 k)$  nejdříve klasickému MergeSortu



3



## Kód studenta 46

### 2 Převod bezkontextové gramatiky na automat (3 body)

1. Uveďte definici bezkontextové gramatiky.
2. Sestrojte bezkontextovou gramatiku  $G$  generující následující jazyk:

$$L = \{a^i b^j \mid i, j \geq 0 \text{ a platí } 2i = 3j\}$$

3. Převeďte gramatiku  $G$  z předchozí části na zásobníkový automat přijímající prázdným zásobníkem. (Pokud se vám nepodařilo gramatiku sestrojit, sestrojte libovolný zásobníkový automat přijímající jazyk  $L$  prázdným zásobníkem.)

1) Bezkontextová gramatika je  $G = (V, T, P, S)$ , kde:

$V$  je množina neterminálů  
 $T$  je množina terminálů

$P$  je množina pravidel tvaru  $A \rightarrow \lambda$ , kde  $A \in V, \lambda \in (V \cup T)^*$

$S$  je počáteční symbol  $S \in V$

✓

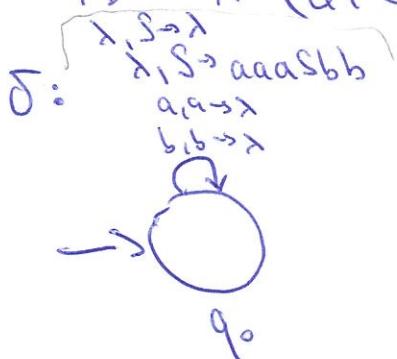
2)  $G = (V, T, P, S) \quad V = \{S\} \quad T = \{a, b\}$

P:

$S \rightarrow a a a S b b b \mid \lambda$

✓

3) PDA  $A = (Q, \Sigma, \Gamma, \delta, q_0)$   $\Sigma = \{a, b\}$   $\Gamma = \{a, b, S\}$   $Q = \{q_0\}$



✓



## Kód studenta 46

### 3 Lineární algebra (3 body)

Uvažujme dvě maticy

$$A = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}.$$

1. Najděte vektor  $x \in \mathbb{R}^2$  takový, aby vektory  $Ax, Bx$  byly na sebe kolmé (při standardním skalárním součinu).
2. Najděte matici lineárního zobrazení, které vektor  $Ax$  zobrazuje na vektor  $Bx$  pro libovolné  $x \in \mathbb{R}^2$ .
3. Rozhodněte, zda lineární zobrazení  $x \mapsto B^{-1}A^3B^{-1}x$  plochy geometrických útvarů zvětšuje či zmenšuje.

1) Chceme  $\langle Ax, Bx \rangle = 0$  tedy  $(Ax)^T Bx = 0$  tedy  $x^T A^T B x = 0$

$$A^T B = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 1 & 4 \end{pmatrix} \quad \text{nech } x = (x_1, x_2)^T$$

$$\begin{pmatrix} 1 & 3 \\ 1 & 4 \end{pmatrix} \cdot x = \begin{pmatrix} x_1 + 3x_2 \\ x_1 + 4x_2 \end{pmatrix} \quad x^T \begin{pmatrix} x_1 + 3x_2 \\ x_1 + 4x_2 \end{pmatrix} = x_1(x_1 + 3x_2) + x_2(x_1 + 4x_2)$$

$$x_1^2 + 3x_1x_2 + x_1x_2 + 4x_2^2 = 0 \quad x_1^2 + 4x_1x_2 + 4x_2^2 = 0$$

→ uvažme jako rovnici  $x_1$

$$D = (4x_2)^2 - 4 \cdot 4x_2^2 = 0 \quad x_1 = \frac{-4x_2}{2} = -2x_2 \quad x = (-2x_2, x_2)^T \checkmark$$

nejdílečné tedy  $(0,0)^T$  ✓

2) Nechť  $y = Ax$  pak  $x = A^{-1}y$  (A je regulérné, má lin. nezávislé řádky).

Zobrazení do  $Bx$  je pak  $f(y) = B A^{-1}y$ , matici lin. zobrazení je  
tedy  $B \cdot A^{-1}$ .  $A^{-1}$  určíme pomocí  $\left( \begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array} \right) \sim \left( \begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 0 & -1 & -1 & 1 \end{array} \right) \sim \left( \begin{array}{cc|cc} 1 & 0 & -1 & 2 \\ 0 & 1 & 1 & -1 \end{array} \right) \sim \left( \begin{array}{cc|cc} 1 & 0 & 1 & 1 \\ 0 & 1 & -1 & 1 \end{array} \right)$

$$A^{-1} = \begin{pmatrix} -1 & 2 \\ 1 & -1 \end{pmatrix}, \quad B \cdot A^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} -1 & 2 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ 1 & 0 \end{pmatrix} \checkmark$$

Uvážme tedy  $A \cdot A^{-1} = I_2$  a  $x \in \mathbb{R}^2$   $f(Ax) = B \cdot A^{-1}Ax = B \cdot \underline{\underline{I_2}}x = Bx$

5) libovolný geometrický úzavír v  $\mathbb{R}^2$  se dá reprezentovat jeho

orientovaným rovnočlenstvem. Poloha body pro libovolný

rovnočlenstvem  $x, y \in \mathbb{R}^2$  je jeho <sup>tone</sup> <sup>poloha</sup> vzhledem k jeho orientaci.

Oblast libovolného rovnočlenství je pak  $\det(x_1 x_2) / \det(y_1 y_2)$  <sup>oblast</sup>

obrazem je  $|\det(B^{-1}A^3B^{-1}(x_1 x_2))| = |\det(B^{-1})^2 \det(A)^3 \det(y_1 y_2)|$

Nyní, že součin det = det součin

Stále lze vidět, že  $(\det(B^{-1})^2 \det(A)^3)$  je  $>1$  /  $<1$  /  $=1$

$\uparrow$  rozšíření  $\uparrow$  zmenšení  $\uparrow$  zmenšení

$$B^{-1} = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix} \quad \det(B^{-1}) = 2 \cdot 0 - 1 = (-1)^2 = 1$$

$$= 0 - 1 = -1$$

$$A = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix} \quad \det(A) = 1 \cdot 1 - 2 \cdot 1 = -1 \quad (-1)^3 = -1$$

$$|1 \cdot (-1)| = 1$$

□

Doch vzhledem

$$B = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}$$

$$\left( \begin{array}{cc|cc} 0 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{array} \right) \sim \left( \begin{array}{cc|cc} 1 & 2 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array} \right) \sim \left( \begin{array}{cc|cc} 1 & 0 & -2 & 1 \\ 0 & 1 & 1 & 0 \end{array} \right)$$

$B$

$I_2$

$I_2$

$B^{-1}$



3+

Kód studenta 46



#### 4 Pravděpodobnost (3 body)

- Napište větu o linearitě střední hodnoty (nedokazujte ji).
- Ve skupině  $n$  lidí každý hodí (férovou) šestistěnnou kostkou. Každá dvojice lidí, kteří hodili stejně číslo, získá jeden bod. Označme  $Y$  počet bodů získaných jednou předem vybranou dvojicí. Určete střední hodnotu veličiny  $Y$ .
- Označme  $X$  celkový počet získaných bodů za všechny dvojice (jeden člověk může skórovat ve více dvojicích). Určete střední hodnotu veličiny  $X$ .

1) Nechť  $X, Y$  jsou náležné veličiny. Pak  $\mathbb{E}(X+Y) = \mathbb{E}(X) + \mathbb{E}(Y)$ . ✓

2) Dany dvojice lidí získají bod, pokud hod prvního  $H_1$  se rovná hodu druhého  $H_2$ . Jinak nezískají nic.

$\mathbb{E}(Y) = 0 \cdot P(H_1 \neq H_2) + 1 \cdot P(H_1 = H_2) = P(H_1 = H_2)$ . Ze všech možných kombinací když hodí je  $6 \times 36$  výhodných když

$$P(H_1 = H_2) = \frac{1}{6} \Rightarrow \mathbb{E}(Y) = \frac{1}{6}$$

— ✓

3) Všechny lidi v dvojice jsou nezávislé. Naleme  $\binom{m}{2}$  dvojice a pro dvojici platí, že  $\mathbb{E}(\text{bod dvojice}) = \frac{1}{6}$ .  $\mathbb{E}(X) = \mathbb{E}\left(\sum_{Y \in \binom{[m]}{2}} \mathbb{E}(Y)\right) =$

$$= \sum_{Y \in \binom{[m]}{2}} \mathbb{E}(\mathbb{E}(Y)) = \sum_{Y \in \binom{[m]}{2}} \mathbb{E}(Y) = \sum_{Y \in \binom{[m]}{2}} \frac{1}{6} = \frac{1}{6} \binom{m}{2} = \frac{1}{12} m \cdot (m-1)$$

Suma  
dvojic

Suma  
dvojic  
Následných  
dvojic  
mij

✓



3



## Kód studenta 46

### 5 Překlad adresy (otázka studijního zaměření – 3 body)

Předpokládejme hypotetický procesor s paměťovou architekturou, kde virtuální i fyzický prostor používá 16-bitové adresování. Pro překlad adres se používá 1-úrovňové stránkování s 64 B stránkami. Záznamy ve stránkovací tabulce (které mají také 16 bitů) využívají offsetové byty pro přidružené příznaky, nejméně významný bit je použit jako příznak platnosti (tj. 1 = záznam je platný, 0 = záznam není platný).

Stránkovací tabulka, jejíž výpis následuje níže, se nachází na adrese 0x1d40. Výpis zobrazuje pro jednoduchost přímo 16-bitové položky stránkovací tabulky, tedy není nutné řešit pořadí bytů v reprezentaci čísel.

	+0x0	+0x2	+0x4	+0x6	+0x8	+0xA	+0xC	+0xE
0x1d40	0xdecd	0x0149	0x21c9	0x2b19	0x8f85	0x65d9	0x68a1	0x2151
0x1d50	0x2fb1	0x8249	0xabc5	0x3219	0xeacd	0x2651	0x9111	0x4249
0x1d60	0x5085	0x8e4d	0xe731	0xb109	0xb751	0x7fe2	0x3d6a	0xcf24
0x1d70	0x9c9c	0x6011	0x3d11	0xd589	0xc14d	0x748d	0xa019	0x41c9
0x1d80	0x0320	0x0e2a	0xd2d6	0xb0a8	0x8b44	0x3aa2	0xe4ce	0xcde0
0x1d90	0x5068	0xfb8	0x427e	0x007c	0x7bba	0x578e	0x5770	0x4f76
0x1da0	0x85b6	0xe920	0x7768	0xcaf0	0xe3c0	0x76fa	0xc84a	0xdc66
0x1db0	0x94d4	0x3fb2	0xb986	0xe52c	0x1ddc	0x957e	0x8ccc	0x82a2
0x1dc0	0x4f3c	0xf6d2	0x8fc2	0xee90	0x9792	0xa43a	0xc590	0x9180
0x1dd0	0x1386	0x2b92	0xdcf0	0x62a0	0x47d8	0x9ede	0x9614	0x8d9c
0x1de0	0xda10	0x16d0	0xc68c	0xa108	0x45c8	0x1f60	0x4a18	0xe9ce
0x1df0	0x4c4f	0xbbbe9	0x9535	0x74eb	0x6b19	0x7aad	0x36b1	0x85fd
0x1e00	0xcfef3	0x7dd3	0x4917	0x54ad	0x9391	0x92c5	0x0ab9	0x9fa7
0x1e10	0xd390	0xddfa	0x6330	0x33a2	0xad22	0xabbe	0xe21a	0x04dc
0x1e20	0x43f8	0x4188	0xfe3c	0xba68	0xb082	0x2040	0x126e	0xdecc
0x1e30	0xc206	0xb9a2	0xbe10	0x0cf8	0x81c8	0x564c	0x8140	0x923a
0x1e40	0x5bda	0xbd12	0xb794	0x910c	0x05f2	0x7636	0x9bd8	0xdede
0x1e50	0x85f2	0x21ca	0xf3c8	0x1bc2	0xde4e	0xa87c	0x9a5e	0xc348
0x1e60	0x80e0	0x37ea	0x39a8	0x7eff	0xd8ca	0x79f4	0x286c	0x7f00
0x1e70	0xb046	0x0fe8	0x5256	0x72e8	0xe0cc	0xf6aa	0x39d8	0x8c4e
0x1e80	0xdd44	0x5088	0x789a	0x68d0	0x1748	0xfb0	0x9130	0x8e9e
0x1e90	0x13bf	0xe44f	0x525f	0x8715	0x5389	0x849f	0xaeel	0xd451
0x1ea0	0x584a	0xe278	0x6776	0x32b0	0x82b0	0x8b1c	0x704c	0x2e8a
0x1eb0	0xabce	0x4020	0xd434	0xaeeea	0x6548	0xdf10	0x3ad8	0x388a
0x1ec0	0xb134	0x8d2c	0xe422	0x6ec8	0x101d	0xa595	0xb55	0x7cf1
0x1ed0	0x69b9	0x431b	0xfb7	0x293d	0x0075	0xd8d7	0x9059	0x1569
0x1ee0	0x4c73	0x8577	0x6efb	0x2e09	0xa58d	0xd858	0x39d2	0x00d8
0x1ef0	0x5eb8	0x92e6	0x5798	0x2314	0xb82c	0x34c2	0xbe16	0xbed6
0x1f00	0x4b12	0x4660	0xb082	0x2eee	0x3ee4	0x9140	0x4b42	0xef9e
0x1f10	0x8318	0x5a74	0x7aa2	0xda8a	0x1504	0x5cf4	0x9876	0x3632
0x1f20	0xb5be	0x802e	0xab94	0x0c46	0xd588	0xb138	0xc378	0x5a8a
0x1f30	0x94ea	0x1254	0x0bac	0xaef2	0x2c6a	0xc5de	0x5b12	0xca64

Přeložte virtuální adresu 0x328f (0b0011001010001111 binárně) na fyzickou adresu:

- Napište číslo stránky a offset pro danou virtuální adresu.
- Napište fyzickou adresu položky ve stránkovací tabulce, která odpovídá zadané virtuální adrese.
- Napište výslednou fyzickou adresu, pokud existuje překlad. Jinak uveďte, že neexistuje.

V odpovědi důsledně uvádějte číselnou soustavu - čísla bez prefixu jsou v desítkové soustavě, prefix 0x označuje šestnáctkovou soustavu, prefix 0b označuje dvojkovou soustavu. (To, jakou soustavu použijete, se ovšem samo o sobě nehodnotí.)

1) Vítmálové adresu  $0x528F = 0b\underbrace{00110010}_{\text{CÍL DO STŘÁVKY}} \underbrace{10001111}_{\text{OFFSET (6b)}} 2^6 = 64$

Cíl do střávky =  $0b0011001010 = 202$  (10b)

offset =  $0b001111 = 15$  ✓

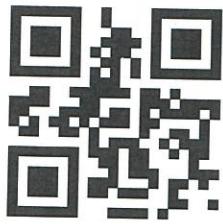
2)  $0x1ED4$  Adresa je leží na pozici  $0x1D40$ , které máme  $2 \times 202$

a dostaneme  $0x1ED4$  ('2\*' protože hodnota mají 9B) ✓

3) Na pozici  $0x1ED4$  leží  $0xFB47 = 0b\underbrace{11111011}_{\text{FRIZCEK ADRESA}} \underbrace{11000111}_{\text{PEČOVÁKY}}$

Nejméně významný bit je 1, tedy základ je fazy. Výsledná adresa je pak  $0b1111101111000111 = 0xFBCF$  (sleplil jsem bytovou adresu a offset) ✓

3 body



## Kód studenta 46

### 6 Komponentový systém (otázka studijního zaměření – 3 body)

Naprogramujte v C++, C# nebo Javě všechny níže zmíněné a další potřebné typy (včetně zmíněných metod) pro implementaci následujícího komponentového herního frameworku. Ve hře vytvořené pomocí vašeho frameworku má každý objekt GO má reprezentovat množinu komponent – kde komponenta je pro nás libovolný vhodný typ (dle vašeho návrhu), jehož instance přidává k instanci GO nějaká silně typovaná data, a/nebo nějakou další funkcionality. Od každého druhu komponenty může být na instanci GO připojena právě 0 nebo 1 instance daného druhu komponenty. Každá instance nějaké komponenty může být připojena na maximálně 1 GO.

1. Na GO by měla být vhodná metoda pro připojení instance nějaké komponenty ke GO – pokud GO komponentu stejného druhu již má, tak nová instance nahradí instanci původní.

Na GO by měla být vhodná metoda pro vrácení instance komponenty zvoleného druhu (požadovaný druh si nějak vhodně může zvolit volající). Pokud na GO žádná instance požadovaného typu komponenty připojená není, tak to má funkce nějak rozumně indikovat (funkci budeme chtít běžně používat pro detekci, zda daný typ komponenty na GO je, nebo není).

Na GO má být metoda Update s parametrem `timeDelta` jako 32-bitové float číslo, které reprezentuje čas v sekundách od předchozího volání metody Update na stejném GO. Metoda Update na GO má projít všechny instance připojených komponent, a pro komponenty, které mají metodu Update se stejným parametrem, se má tato zavolat. Pořadí volání metod Update na komponentách GO není důležité. Ve svém objektovém návrhu vhodně zohledněte, že autor komponenty má mít možnost se rozhodnout, zda komponenta bude metodu Update mít nebo nikoliv.

2. Napište implementaci komponenty Position, která ke GO přidává 32-bitové float souřadnice X a Y reprezentující polohu GO na 2D herní ploše. Napište implementaci komponenty Health, která ke GO přidává 32-bitové float hodnoty maximálního a aktuálního zdraví (aktuální zdraví je vždy v rozsahu 0 až max včetně).

Napište příklad kódu, který vytvoří instanci GO pro kámen, který má komponentu Position, a instanci GO pro trola, který má komponenty Position a Health (max = 30, aktuální = 20).

3. Doplňte vhodné komponenty reprezentující efekt otravy a efekt krvácení – oba efekty vyžadují, aby GO, ke kterému jsou připojené, měl komponentu Health (pokud GO takovou komponentu nemá, tak efekt nic nedělá). Dále jsou oba efekty parametrizované 32-bit float hodnotou, která reprezentuje, kolik aktuálního zdraví efekt odečítá každou sekundu. Pokud je efekt připojený k nějakému GO, tak při zavolání Update na GO má dojít k úpravě zdraví GO dle parametru efektu. Na jednom GO může být maximálně 1 efekt otravy a nezávisle maximálně 1 efekt krvácení (tedy GO může být bez efektu, nebo mít 1 otravu a žádné krvácení, nebo žádnou otravu a 1 krvácení, nebo 1 otravu a 1 krvácení). Do budoucna budeme přidávat další podobné efekty.

Napište implementaci funkce, která dostane seznam GO a hodnotu zranění za sekundu – funkce projde všechny předané GO a pokud GO má komponentu Health, tak ke GO připojí novou instanci efektu otravy nebo krvácení parametrizované předanou hodnotou zranění za sekundu. Volající musí mít navíc možnost si nějak vhodně zvolit, zda funkce aplikuje na předané GO efekt otravy nebo efekt krvácení.

Napište příklad kódu, který na výše vytvořený kámen a trolu zkusí aplikovat efekt krvácení s hodnotou zranění 1.2 za sekundu.

Takovéto se uchází na 2 papírech (dalších) .

K60 STUDENTA 46

07A2KA 6

```
abstract class Component {  
    public static string Name {get; };  
    private GameObject? go {get; init} = null;  
    abstract Component Clone(GameObject go);  
}
```

```
abstract class UpdatableComponent : Component {  
    abstract void Update(float timeDelta);  
}
```

```
abstract class NegativeEffectComponent : UpdatableComponent {  
    private Health? h = null;
```

```
public float DPS {get; set;};  
override Update (float timeDelta) {  
    if (go != null) {  
        if (h != null) {  
            h.HP -= DPS * timeDelta;  
        }  
    }  
}
```

else {  
 if (go != null) {  
 if (targetComponent(c, health, Name)) {  
 h = c; // c neni typu Health  
 h.HP -= DPS \* timeDelta;  
 }  
 }  
}

```
class Health : Component {  
    public static Name {get; } = "Health";  
    float maxHealth, {get; set;} = 0;  
    float curvHealth {get; set(x => {  
        if (x < 0) curvHealth = 0;  
        else if (x > maxHealth) curvHealth = maxHealth;  
        else curvHealth = x;  
    }) = 0;}
```

override Component Clone(GameObject go) {

return new Health() {maxHealth =  
maxHealth, curvHealth = curvHealth,  
go = go}; }

?

Health

```
class Position : Component {  
    public static Name {get; } = "Position";  
    float x {get; set;} = 0;  
    float y {get; set;} = 0;  
    override Component Clone(GameObject go) {  
        return new Position() {x = x, y = y, go = go};  
    }
```

```

public class GameObject {
    private UnorderedMap<string, Component> components;
    public GameObject() { components = new(); }

    public bool TryGetComponent (out Component c, string name) {
        c = null;
        if (components.Contains(name)) {
            c = components[name];
            return true;
        }
        return false;
    }

    public void SetComponent (Component c) {
        components[c.name] = c.Clone(this);
    }

    public void Update (float timeDelta) {
        foreach (var c in components) {
            if (c is UpdatableComponent uc) {
                uc.Update (timeDelta);
            }
        }
    }
}

```

silně typováno / bylo lepší

asi nebylo nutné klonovat  
aha - jeh kvůli AddNegEff

GameObject Rock = new();
Position p = new Position() {X=42, Y=10};

Rock.AddComponent (p);

GameObject Troll = new();

Health h = new() { maxHealth = 30, currentHealth = 20};

p.X = 139; p.Y = 3,1415;

Troll.AddComponent (h);

Troll. — " — (p);

Ko's Structure 46

07420A 6

class Poison : NegativeEffectComponent {

    static public string Name {get;} = "Poison";

    override Component Clone(GameObject go) {

        //和睦 clone

}

}

class Bleeding : NegativeEffectComponent {

    static public Name {get;} = "Bleeding";

    override Component Clone(GameObject go) {

        //deep clone

}

}

AddNegativeEffect(List<GameObject> objects, NegativeEffectComponent nec, float DPS) {

nec.DPS = DPS;

foreach (var o in objects) {

    if (o.TryGetComponent(Health.Name)) {

        o.AddComponent(nec);

}

}

{ entities

    List<GameObject> = {Rock, Troll};

    AddNegativeEffect(entities, new Bleeding(), 1.2);

13 (3)



3kvp



## Kód studenta 46

### 7 Architektury databázových systémů (otázka studijního zaměření – 3 body)

Uvažujte tabulky PROJECT a EMPLOYEE se schématy PROJECT(ID, Name, Budget) a EMPLOYEE(ID, Name, Position, Salary, PID), kde  $PID \subseteq PROJECT.ID$ .

- 1 – Je uvedené schéma ve 3NF, pokud víte, že firemní politika vyžaduje funkční závislost EMPLOYEE.Position → EMPLOYEE.Salary?

Pokud ano, zdůvodněte. Pokud ne, upravte schéma, aby bylo ve 3NF.

- 2 – Pro výsledné relační schéma nakreslete ekvivalentní konceptuální model pomocí jazyka E-R, případně UML.
- 3 – Napište nad uvedeným schématem v jazyce SQL dotaz, který vrátí seznam projektů i s počty zaměstnanců, kteří na

1) Nem. EMPLOYEE.Position nemá žádoucí funkční závislost. Nedoporučuje 3NF.  
a EMPLOYEE.Salary nemá žádoucí funkční závislost  
PROJECT se nezmění, EMPLOYEE (ID, Name, Position, PID), třídy  
SALARY (Position, Salary), pokudž funkce jednoznačně vrátí PID. ✓

2) UML



3) SELECT PROJECT.ID, PROJECT.Name, PROJECT.Budget, COUNT(\*)  
From PROJECT INNERJOIN EMPLOYEE ON (PROJECT.ID = PID)  
GROUP BY PROJECT.ID, Project.Name

2+



## Kód studenta 46

### 8 Web chat (otázka studijního zaměření – 3 body)

Uvažme jednoduchou webovou aplikaci s přihlašováním, ve které uživatelé sdílejí krátké textové zprávy (chat). Přihlašovací formulář v HTML vypadá přibližně takto:

```
<form action="?action=login" method="POST">
    Login: <input type="text" name="login">
    Password: <input type="password" name="password">
    <button type="submit">Sign in</button>
</form>
```

Front controller webové aplikace (jeho nejpodstatnější části) vypadají následovně.

```
if (($_GET['action'] ?? '') === 'login') {
    if (verify_credentials($_POST['login'], $_POST['password'])) {
        $res = $db->query("SELECT id FROM user WHERE login = '" . $_POST['login'] . "'");
        $userId = $res->fetch_column(0);
        if ($userId) setcookie('user', $userId);
    } else {
        showWrongCredentialsError();
    }
    redirectAndExit();
}
// ...
if (empty($_COOKIE['user'])) {
    showLoginForm();
} else {
    showChatMessagesForUser($db, $_COOKIE['user']);
}
```

SQL injection  
↳ 1

výsledek je vlastní  
slouží cookies  
↳ 2

```
// ... v pomocných inkludovaných souborech se také nachází
function showChatMessagesForUser($db, $userId) {
    // ...
    $messages = loadMessagesForUser($db, $userId);
    foreach ($messages as $m) {
        echo "<p>[$m->from]: $m->text</p>";
    }
    // ...
}
```

minulé kód script  
↳ 3

Můžete předpokládat, že chybějící části kódu (včetně funkcí) jsou rozumně implementovány (a neobsahují bezpečnostní chyby). Např. proměnná \$db obsahuje Mysqli objekt reprezentující spojení k databázi, superglobální proměnné nebyly skriptem modifikovány atd.

Identifikujte všechny bezpečnostní chyby ve výše uvedených příkladech. U každé chyby stručně popište možný způsob zneužití (v případě injection útoků uveďte zejména, jaký řetězec by byl injektován) a způsob opravy (načrtněte krátký kód, nebo stručně popište, jaké změny je třeba v kódu provést).

Pokud v kódu žádné zranitelnosti nejsou, popište stručně jeden možný typ injection útoku (na tuto konkrétní aplikaci) a identifikujte bezpečnostní mechanismy v kódu, které tomuto útoku zabrání.

1) Typický příklad SQL injection. Polníček "login" nemá nijak omezený, můžete vložit například "SELECT \* FROM user WHERE login = OR cond;" kde cond bude jenom řetězec, který plní podmínku a my se dostaneme do chasu někoho jiného. - Ne, login + heslo se předtím testuje, takže je tady problém.

Jindy příklad je, že napíšete řetězec "DROP TABLE table-name;", kde table-name plníste svým uživateli, tedy "user". Tímto smazete tabulku z databáze! ?? Table se to nedělá

Zde je řešení využít funkci, kterou kontrolují řetězec (z SQL a zdrojového code). Případně validovat vstupní data (jistě k tomu návíc) - sanitizace paranců parametrických SQL dotazů.

2) Polníček si můžete na své stránce nastavit cookies. Např. user = id, kde id slouží uživateli a hodnotu se mu lze způsobit něčím chybou.

Jako?

Zde je potřeba být všechno vědět, aby se mohlo ověřit aneb si uložit data do session, ale session používají tedy cookies, den je rodi? Kde, když je to už uživatel nebo klient? ans, ale session používají tedy cookies, den je rodi? Kde, když je to už uživatel nebo klient?

Následně můžeme použít cookies, ale musíme zvolit id kde, aby se berlēpodobně jeho vložené normálně neodpovídající vložky vložily do session, ale ID je obvykle součástí href, nemá to být věc.

Jako?

3) Polníček někrom poslal referenci, kde bude nějakej javascript nebo HTML, kde se spustí. Příklad text="</p><script type=javascript>...problematika</script></p>". Zde je potřeba escapovat vložený řetězec, když je ...<script></p>. Zde je potřeba escapovat vložený řetězec, když je '<' nebo '>' ale nelze je zvládat jde o definici tagu.