B **Jméno** _____

1. **(2 body, více správných odpovědí, 1 chyba = 1 bod, více chyb = 0 bodů)**
   V jaké třídě složitosti je funkce `addOne` v následujícím programu vzhledem k velikosti pole `p` (budeme ji značit n)? Předpokládejte, že funkce `new` pracuje v konstantním čase vzhledem k velikosti pole `p`. Dále předpokládejte, že velikost hodnoty proměnné `N` vždy před a po provedení funkce `addOne` koresponduje s velikosti pole `p`.

```
public class MyCounter {

     int N = 0;
     int [] p;

     public void addOne (void)
     {
          int i=0;
          while ((i < N) && (p[i] == 3)) {
               p[i] = 0;
               i++;
          }
          if (i==N) {
               int m = N+1;
               p = new int[m];
               for (int j = 0; j<m; j++) p[j]=0;
               N = m;
          }
          p[i] += 1;
     }
}
```

a) $O(1)$
b) $O(n \cdot \log(n))$
c) $O(n^2)$
d) $\Omega(1)$
e) $\Omega(\log(n))$
f) $\Omega(n^2)$
g) $\Theta(1)$
h) $\Theta(n)$
i) $\Theta(n^2)$

2. **(2 body, jedna správná odpověď)**
   Funkce:

```
int foo(int x, int y) {
 if (x == y) return 0;
 if (x < y) return foo(y-1,x)+1;
 else return foo(x-1,y)+1;
}
```

a) buď hned vrátí první parametr nebo jen „do nekonečna“ volá sama sebe.
b) vrátí absolutní hodnotu rozdílu obou parametrů.
c) vrátí součet svých parametrů.
d) vrátí maximální hodnotu z obou parametrů.
e) neprovede ani jednu z předchozích možností.

3. (2 body, jedna správná odpověď)
Funkce $K(n,m)$ je definována níže. Vypočtěte ručně hodnotu $K(2,1)$.

$$K(n,m) = \begin{cases} m+1 & \text{pro } n = 0, \\ K(n-1,1)+n & \text{pro } n > 0 \text{ a } m = 0, \\ K(n-1,K(n,m-1)) & \text{pro } n > 0 \text{ a } m > 0. \end{cases}$$

a) 2
b) 5
c) 6
d) 7
e) 8
f) 9
g) hodnota není definovaná

4. (2 body, více správných odpovědí, 1 chyba = 1 bod, více chyb = 0 bodů)
Určete, do jaké třídy složitosti náleží následující rekurentně definovaná funkce T:

$$T(n) = 100T\left(\frac{n}{10}\right) + n$$

a) $T(n) \in O(n \cdot \log(n))$
b) $T(n) \in O(n^2)$
c) $T(n) \in O(n^3)$
d) $T(n) \in O(n^n)$
e) $T(n) \in \Omega(n^2)$
f) $T(n) \in \Omega(n^3)$
g) $T(n) \in \Omega(n)$
h) $T(n) \in \Omega(n \cdot \log(n))$
i) $T(n) \in \Theta(n \cdot \log(n))$
j) $T(n) \in \Theta(n^2 \cdot \log(n))$
k) $T(n) \in \Theta(n^2)$
l) $T(n) \in \Theta(n^3)$

5. (1 bod, jedna správná odpověď)
Hashovací (nebo také rozptylovací) funkce:
a) pro daný klíč vypočte adresu
b) převádí adresu daného prvku na jemu příslušný klíč
c) vrací pro každý klíč jedinečnou hodnotu
d) vrací pro dva stejné klíče různou hodnotu

6. (1 bod, jedna správná odpověď)
Metoda univerzálního hashování:
a) dokáže snížit počet kolizí změnou hashovaní funkce v průběhu hashování
b) dokáže uložit libovolný předem neznámý počet klíčů
c) nemá problém s kolizemi, protože nevznikají
d) používá pouze jedinou tzv. univerzální hashovaní funkci

7. (1 bod, jedna správná odpověď)
V otevřeném hashování (open-address hashing):
a) je nutno definovat rozsah hodnot klíčů
b) je nutno po určitém počtu kolizí zvětšit velikost pole
c) je možno uložit libovolný počet synonym
d) je počet uložených prvků omezen velikostí pole

8. (1 bod, jedna správná odpověď)

Metoda hashování se separovanými řetězci (chaining)

a) dokáže uložit libovolný předem neznámý počet klíčů
b) nemá problém s kolizemi, protože nevznikají
c) řeší kolize uložením klíče na první volné místo v poli
d) dokáže uložit pouze předem známý počet klíčů

9. (2 body, jedna správná odpověď)

Jak vypadá hashovací tabulka po vložení následujících klíčů A, B, C, D, E, F, G (v tomto pořadí) metodou EISCH (early insert standard coalesced hashing), když známe hodnoty hashovací funkce *h* (viz níže) a velikost celé hashovací tabulky (M'=10)?

h(A)=1, h(B)=3, h(C)=1, h(D)=4, h(E)=3, h(F)=10, h(G)=1

a)

1	A	7
2		
3	B	9
4	D	
5		
6		
7	G	10
8	F	
9	E	
10	C	8

b)

1	G	7
2		
3	E	9
4	D	
5		
6		
7	C	10
8	A	
9	B	
10	F	8

c)

1	A	7
2		
3	B	9
4	D	
5		
6		
7	G	8
8	F	10
9	E	
10	C	

d)

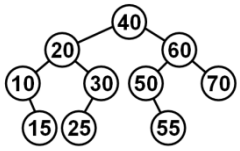
1	A	10
2		
3	B	9
4	D	
5		
6		
7	G	
8	F	7
9	E	
10	C	8

10. (1 bod, jedna správná odpověď)

Dynamické programování je metoda, která

a) umožňuje řešit všechny problémy jako metoda rozděl a panuj, ale s lepší nebo stejnou asymptotickou paměťovou složitostí
b) umožňuje řešit problémy pomocí dynamické alokace datových struktur v paměti
c) umožňuje efektivně řešit úlohy s různě velkými vstupy
d) umožňuje efektivně řešit problémy pomocí rozkladu na podproblémy, které sdílejí společný podprostor řešení.

11. (2 body, jedna správná odpověď)
Na obrázku je uveden BVS, který v průběhu práce vyvažujeme (AVL strom). Ten nyní upravíme tak, že z něj odstraníme operací Delete uzly s klíči 50, 30, 25 v tomto pořadí. Rozhodněte, zda a jaká rotace bude během této úpravy použita:



a) L rotace
b) R rotace
c) LR rotace
d) RL rotace
e) žádná rotace

12. (2 body, jedna správná odpověď)
Pravidelný binární strom má hloubku k (připomínáme, že kořen má hloubku 0). Počet listů tohoto stromu leží právě v intervalu

a) $< 2, 2^k - 1>$
b) $< \log_2 k, k>$
c) $< \log_2 k, k+1>$
d) $<k, 2^k>$
e) $<k+1, 2^k>$

13. (1 bod, jedna správná odpověď)
Pole délky n obsahuje hodnoty n, 1, 2, 3, 4, ... , $n-2$, $n-1$, v uvedeném pořadí. Počet porovnání dvojic čísel, které provede Insert sort při řazení tohoto pole do neklesajícího pořadí, je

a) $2 \cdot \log_2(n) - 1$
b) $n+3$
c) $2n - 3$
d) $n(n-1)/2 + 2$
e) $2n^2 - n - 1$

14. (1 bod, jedna správná odpověď)
O řazení sléváním, běžně známém jako Merge sort, lze s určitostí říci:

a) je stabilní, protože jeho složitost je $\Theta(n \cdot \log(n))$
b) je nestabilní, protože jeho složitost je $\Theta(n \cdot \log(n))$
c) je rychlé (tj. má složitost $\Theta(n \cdot \log(n))$) právě proto, že je stabilní
d) lze je napsat tak, aby nebylo stabilní
e) je vždy nestabilní

15. (2 body, jedna správná odpověď)
Na vstupu Heap sortu je neseřazené pole uvedené níže.
27 18 21 19 13 16 26 17 23
V první fázi řazení Heap sort vytvoří haldu s nejmenším prvkem na začátku pole. Tato halda je

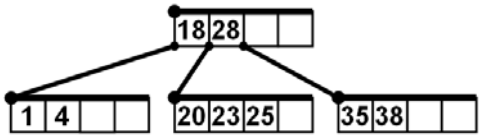
a) 13 16 21 18 19 17 23 26 27
b) 13 16 17 27 18 21 26 19 23
c) 13 18 19 16 17 21 26 27 23
d) 13 17 16 23 18 21 26 19 27
e) 13 17 16 19 18 21 26 27 23

16. (1 bod, jedna správná odpověď)
Vstupní pole pro řazení obsahuje hodnoty v uvedeném pořadí:
6 6 9 4 6 11 4 6 9 9
Pole řadíme pomocí Counting sortu. Toto řazení používá pomocné pole, které se nejprve plní četnostmi vstupních hodnot a pak se ještě dále zpracovává. Těsně předtím, než se začne plnit výstupní pole (které je indexováno od 0), je obsah pomocného pole následující

a) od indexu 0 do indexu 9: 4 4 5 6 7 8 9 9 10 11
b) od indexu 3 do indexu 12 : 11 9 9 9 6 6 6 6 4 4
c) od indexu 3 do indexu 10: 2 2 6 6 6 9 9 10
d) od indexu 4 do indexu 11: 1 0 3 0 0 4 0 2
e) od indexu 4 do indexu 11: 1 1 5 5 5 8 8 9

17. (2 body, jedna správná odpověď)
Z B-stromu znázorněného na obrázku odebereme postupně klíče 4, 35. Pak bude kořen obsahovat klíč/klíče

a) 18
b) 18, 28
c) 20
d) 20, 28
e) 28



18. (2 body, jedna správná odpověď)
Jednotlivé klíče binárního stromu vypíšeme nejprve v pořadí Inorder a potom v pořadí procházení do šířky. Získáme tak posloupnosti (B, A, C, D, E, F, G), a (E, B, F, C, G, A, D). Po vypsání klíčů v pořadí Postorder získáme posloupnost

a) (A, B, C, B, G, F, E)
b) (A, D, C, B, G, F, E)
c) (G, B, D, A, C, F, E)
d) (G, B, A, D, F, C, E)
e) (A, B, C, D, G, F, E)

19. (1 bod, jedna správná odpověď)
Quick sort řadí následující šestiprvkové pole čísel.
8 6 2 3 9 4
Jako pivotní hodnotu volí první v pořadí tj. nejlevější. Jak bude pole rozděleno na „malé“ a „velké“ hodnoty po jednom průchodu polem? (Lomítko naznačuje místo dělení.)

a) **3 4 2 / 6 8 9**
b) **2 3 4 / 6 8 9**
c) **4 6 / 2 3 9 8**
d) **4 6 2 / 3 9 8**
e) **4 6 2 3 / 9 8**

20. (1 bod, jedna správná odpověď)
Radix sort řadí pole řetězců {abcd, dcba, aabc, ddba, abbc, cbad, dacb, bbbb, dcca }.
Po prvních dvou průchodech algoritmu bude seznam odpovídající znaku “c” obsahovat právě řetězce v uvedeném pořadí

a) aaba, dabc, abcd
b) cbad, abbc, abcd
c) dcca, dacb, abcd
d) aabc, abbc
e) seznam bude prázdný