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

2. (2 body, jedna správná odpověď)

Funkce:

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

a) buď hned vrátí první parametr nebo jen „do nekonečna“ volá sama sebe.
b) vrátí x+1.
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 $K(1,2)$.

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

a) 2
b) 3
c) 4
d) 5
e) 6
f) 7
g) 8

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:

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

a) $T(n) \in O(n^2)$
b) $T(n) \in O(n \cdot \log(n))$
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) $T(n) \in \Theta(n^2)$
j) $T(n) \in \Theta(n \cdot \log(n))$
k) $T(n) \in \Theta(n^2 \cdot \log(n))$
l) $T(n) \in \Theta(n^3)$

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

6. (1 bod, jedna správná odpověď)
Metoda otevřeného hashování (open-address hashing):
a) dokáže uložit libovolný předem neznámý počet klíčů
b) nemá problém s kolizemi, protože nevznikají
c) ukládá prvky s klíči v dynamické paměti
d) ukládá prvky do pole pevné délky

7. (1 bod, jedna správná odpověď)
Metoda otevřeného hashování s dvojitým hashováním (double hashing):
a) má stejnou pravděpodobnost vzniku dlouhých clusterů jako metoda otevřeného hashování s lineárním přidáváním (linear probing)
b) je metoda ukládání klíčů na dvě různá místa
c) je metoda zmenšující délku clusterů u otevřeného hashování
d) má vyšší pravděpodobnost vzniku dlouhých clusterů než metoda otevřeného hashování s lineárním přidáváním (linear probing)

8. (1 bod, jedna správná odpověď)

Metoda hashování se separovanými řetězci (chaining)

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

9. (2 body, jedna správná odpověď)

Jak vypadá hashovací tabulka po vložení následujících klíčů A, B, C, D, E, F, G, H (v tomto pořadí) metodou EICH (early insert coalesced hashing), když známe hodnoty hashovací funkce *h* (viz níže), velikost celé hashovací tabulky (M'=11) a velikost sklepa (K=2)?

h(A)=1, h(B)=3, h(C)=1, h(D)=1, h(E)=3, h(F)=1, h(G)=8, h(H)=1

a)

1	A	6
2		
3	B	9
4		
5		
6	H	8
7	G	10
8	F	7
9	E	
10	D	11
11	C	

b)

1	A	10
2		
3	B	9
4		
5		
6	H	8
7	G	
8	F	7
9	E	
10	D	11
11	C	6

c)

1	A	11
2		
3	B	9
4		
5		
6	H	
7	G	6
8	F	7
9	E	
10	D	8
11	C	10

d)

1	A	11
2		
3	B	9
4		
5		
6	H	7
7	G	8
8	F	9
9	E	
10	D	6
11	C	10

10. (1 bod, jedna správná odpověď)

Dynamické programování je metoda, která

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

11. (2 body, jedna správná odpověď)
Jednotlivé klíče binárního stromu vypíšeme nejprve v pořadí Inorder a potom v pořadí procházení do šířky. Získáme tak posloupnosti (A, F, B, D, C, G, E), a (G, F, E, A, D, B, C). Po vypsání klíčů v pořadí Postorder získáme posloupnost

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

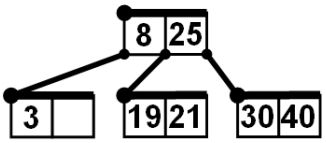
12. (2 body, jedna správná odpověď)
Do prázdného BVS, který v průběhu práce vyvažujeme (AVL strom), vložíme klíče 25, 30, 20, 17, 31, 24, 11, 12. Po prvním rozvážení stromu je třeba provést rotaci, která strom opět vyváží. Tato rotace bude:

a) L
b) R
c) LR
d) RL
e) žádná, strom se v průběhu vkládání daných klíčů nerozváží

13. (2 body, jedna správná odpověď)
Binární strom má právě 5 listů a všechny leží v hloubce 3. Možný počet vnitřních uzlů tohoto stromu (včetně kořene, který má hloubku 0) je právě

a) 2 až 5
b) 3 až 6
c) 5 až 8
d) 6 až 7
e) pouze 7

14. (2 body, jedna správná odpověď)
Do B-stromu znázorněného na obrázku vložíme postupně klíče 7, 5. Pak bude kořen obsahovat klíč/klíče



a) 5
b) 5, 7
c) 7
d) 7, 8
e) 8

15. (1 bod, jedna správná odpověď)
Pomocí Insert sortu řadíme pole délky n o do neklesajícího pořadí. První třetina a třetí třetina pole obsahují čísla v rozmezí 100 až 1000, druhá třetina pole obsahuje čísla v rozmezí 20 až 50. Čas potřebný na seřazení tohoto pole je

a) konstantní
b) úměrný n
c) úměrný $n \cdot \log(n)$
d) úměrný n^2
e) úměrný n^3

16. (1 bod, jedna správná odpověď)
V určitém problému je velikost zpracovávaného pole s daty rovna rovna $n^3 + n$, kde n charakterizuje velikost problému. Pole se řadí pomocí Merge sort-u. Asymptotická složitost tohoto řazení v závislosti na hodnotě n je

a) $\Theta(n \cdot \log(n))$
b) $\Theta(n^3 + n)$
c) $\Theta(n^3 \cdot \log(n))$
d) $\Theta(n^3 \cdot \log^2(n))$
e) $\Theta((n^3 + n)^6)$

17. (1 bod, jedna správná odpověď)
Quick sort řadí následující šestiprvkové pole čísel.

6 2 7 9 5 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) **2 4 / 5 6 7 9**
b) **2 4 5 6 7 / 9**
c) **4 2 / 5 6 9 7**
d) **4 2 5 / 9 7 6**
e) **6 2 4 / 5 9 7**

18. (2 body, jedna správná odpověď)
Následující posloupnost čísel představuje haldu uloženou v poli.

15 18 32 40 52 51 45 80 62 60

Heap Sort (řazení haldou) řadí pole do nerostoucího pořadí. Proveďte první krok druhé fáze řazení, tj.
a) zařaďte nejmenší prvek haldy na jeho definitivní místo v poli a
b) opravte zbytek tak, aby opět tvořil haldu.

a) 15 18 32 40 45 51 52 60 62 80
b) 18 32 40 62 52 51 45 80 60 15
c) 18 32 62 40 52 51 45 80 60 15
d) 18 40 32 60 52 51 45 80 62 15
e) 32 51 45 18 40 52 80 62 60 15

19. (1 bod, jedna správná odpověď)
Radix sort řadí pole řetězců {ttus, rrst, uurs, ttrt, urut, sutr, tttr, stts, trus}.
Po prvních dvou průchodech algoritmu bude seznam odpovídající znaku “t” obsahovat právě řetězce v uvedeném pořadí

a) rrst, ttrt, urut
b) stts, sutr, tttr
c) sutr, tttr, stts
d) ttus, ttrt, tttr, stts
e) ttus, ttrt, tttr, trus

20. (1 bod, jedna správná odpověď)
Řadíme pole celých čísel pomocí Counting sortu. Těsně předtím, než se začne plnit výstupní pole, je obsah změněného (ve 2. kroku metody) pole četností následující (slabě psaná čísla jsou indexy):

14 15 16 17 18 19 20
1 1 5 6 6 6 8

Výstupní pole je indexováno od 0, jeho obsah je

a) 1, 5, 6, 8
b) 14, 15, 16, 16, 16, 16, 17, 18, 19, 20, 20
c) 14, 14, 16, 16, 16, 16, 17, 20, 20
d) 14, 16, 17, 19, 20
e) 14, 16, 17, 20