A Jméno _____

1. (2 body, více správných odpovědí, 1 chyba = 1 bod, více chyb = 0 bodů)

V jaké třídě složitosti je funkce `push` v následujícím programu vzhledem k velikosti pole `p` (velikost budeme 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 `push` koresponduje s velikosti pole `p`. Proměnná `i` je vždy v intervalu `-1` až `N`.

```
public class MyStack {

     int N = 0;
     int i = -1;
     int [] p;

     public void push (int key)
     {
          if (++i >= N) {
               int m = N*2+1;
               int [] t = new int[m];
               for(int j = 0; j<N; j++, t[j] = p[j]);
               N = m;
               p = t;
          }
          p[i] = key;
   }
}
```

a) O(1)
b) O(n*log(n))
c) O(n²)
d) Ω(1)
e) Ω(log(n))
f) Ω(n²)
g) Θ(n²)
h) Θ(n)
i) Θ(1)

2. (2 body, jedna správná odpověď)

Funkce:

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

a) sečte x a y, je-li y nezáporné.
b) pro kladná y vrátí y, jinak vrátí x.
c) spočte rozdíl x–y, je-li y nezáporné.
d) spočte rozdíl y–x, je-li y nezáporné.
e) vrátí hodnotu svého většího parametru.

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

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

a) 0
b) 1
c) 2
d) 3
e) 4
f) 5
g) 6

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:

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

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)
Kolize u hashovací (nebo také rozptylovací) funkce $h(k)$:
a) je situace, kdy pro dva různé klíče k vrátí $h(k)$ stejnou hodnotu
b) je situace, kdy pro dva stejné klíče k vrátí $h(k)$ různou hodnotu
c) je situace, kdy funkce $h(k)$ při výpočtu havaruje (skončí chybou nebo výjimkou)
d) je situace, kdy v otevřeném hashování (open-address hashing) dojde dynamická paměť

6. (1 bod, jedna správná odpověď)
Metoda perfektního hashování (perfect 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) vždy ukládá prvky do lineárního spojového seznamu

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

8. (1 bod, jedna správná odpověď)

Metoda hashování se separovanými řetězci (chaining)

a) nemá problém s kolizemi, protože při ní nevznikají

b) dokáže uložit pouze předem známý počet klíčů

c) ukládá synonyma do samostatných seznamů v dynamické paměti

d) ukládá synonyma spolu s ostatními klíči v poli

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 VICH (variable 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 průběžné dynamické alokace řešení podproblémů v paměti

b) umožňuje řešit pouze úlohy s různě velkými vstupy

c) umožňuje efektivně řešit problémy pomocí rozkladu na podproblémy, které jsou nezávislé (nesdílejí společný podprostor řešení).

d) umožňuje řešit všechny problémy jako metoda rozděl a panuj, ale s lepší nebo stejnou asymptotickou časovou 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, B, E, C, F, G, D), a (C, B, D, A, E, F, G). Po vypsání klíčů v pořadí Preorder získáme posloupnost

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

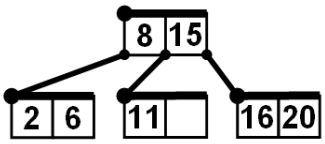
12. (2 body, jedna správná odpověď)
Do prázdného BVS, který v průběhu práce vyvažujeme (AVL strom), vložíme klíče 23, 11, 24, 31, 17, 20, 30, 25. Po prvním rozvážení stromu je třeba provést rotaci, která strom opět vyváží. Tato rotace bude:

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 hloubku 3 (hloubka kořene je 0) a 4 vnitřní uzly. Možný počet listů tohoto stromu je právě

a) 1 až 4
b) 1 až 5
c) 2 až 5
d) 2 až 6
e) 3 až 6

14. (2 body, jedna správná odpověď)
Do B-stromu znázorněného na obrázku vložíme postupně klíče 14, 10. Pak bude kořen obsahovat klíč/klíče



a) 8, 10
b) 8, 11
c) 10
d) 11
e) 11, 14

15. (1 bod, jedna správná odpověď)
Pole délky n obsahuje všechna čísla 1, 2, ..., n, přitom první polovina pole obsahuje všechna lichá čísla seřazená vzestupně, druhá polovina obsahuje všechna sudá čísla také seřazená vzestupně. Pomocí Insert sortu řadíme toto pole do vzestupného pořadí. 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 $n^2 \cdot \sqrt{n}$,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) $\Theta(n^2 \cdot \sqrt{n})$
c) $\Theta(n \cdot \sqrt{n} \cdot \log(n))$
d) $\Theta(n^2 \cdot \sqrt{n} \cdot \log(n))$
e) $\Theta(n^5)$

17. (1 bod, jedna správná odpověď)

Quick sort řadí následující šestiprvkové pole čísel.

4 6 1 3 8 5

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) **1 3 4 5 / 6 8**
b) **1 / 3 4 6 8 5**
c) **3 1 / 6 4 8 5**
d) **3 1 6 / 4 8 5**
e) **3 4 1 6 / 8 5**

18. (2 body, jedna správná odpověď)

Následující posloupnost čísel představuje haldu uloženou v poli.

10 23 11 40 35 20 15 42 43 44

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) 10 11 15 20 23 35 40 42 43 43
b) 11 15 20 23 44 40 35 42 43 10
c) 11 15 23 35 40 44 20 10 43 44
d) 11 23 15 40 35 20 44 42 43 10
e) 23 11 40 35 20 15 42 43 44 10

19. (1 bod, jedna správná odpověď)

Radix sort řadí pole řetězců {ddad, babd, ddca, aaba, bcad, dabc, bbab, acdd, cbdc}.
Po prvních dvou průchodech algoritmu bude seznam odpovídající znaku “b” obsahovat právě řetězce v uvedeném pořadí

a) aaba, dabc, babd
b) bbab, abdc
c) dabc, babd, aaba
d) babd, bbab, bcad
e) aaba, abdc, babd, bcab

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):

12	13	14	15	16	17	18	19	20	21
1	**1**	**1**	**1**	**1**	**4**	**6**	**6**	**6**	**7**

Výstupní pole je indexováno od 0, jeho obsah je

a) 1, 4, 6, 7
b) 12, 12, 17, 17, 17, 18, 18, 21
c) 12, 13, 14, 15, 16, 17, 17, 17, 18, 19, 20, 21, 21
d) 12, 17, 18, 21
e) 12, 17, 20, 21