E1. Do třídy `List` dopište kód metod `remove` a `removeAll`. Dejte si pozor, abyste správně ošetřili všechny krajní případy!

```
class Node {
    Node prev, next;
    int contents;

    Node() {
        next = this;
        prev = this;
    }

    Node(int c, Node p) {
        contents = c;
        next = p.next;
        prev = p;
        p.next = this;
        next.prev = this;
    }
}

class List {
    Node first = new Node();

    void add(int c) {
        new Node(c, first);
    }

    void remove(int c) {
        // odstrani jeden vyskyt daneho cisla
    }

    void removeAll(int c) {
        // odstrani vsechny vyskyty daneho cisla
    }
}
```

E2. Z definice dokažte: $f_1 \in O(g_1) \wedge f_2 \in O(g_2) \Rightarrow f_1 + f_2 \in O(g_1 + g_2)$.

E3. Spočítejte asymptotickou časovou složitost funkce `f` v závislosti na `n`.

```
void f(int n) {
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++) {
            array[i][j]++;
            if (i <= j) for (int k = 0; k < n; k++)
                array[i][j] += k;
        }
}
```

E4. Nakreslete, jak bude vypadat binární halda uložená v poli po vykonání následující posloupnosti operací: $insert(5), insert(3), insert(4), removeMin(), insert(7), insert(1), insert(2), removeMin()$.

E5. V implementaci metody `sort` je chyba. Vyznačte kde a vysvětlete proč.

```
void sort(int[] array) {
    for (int i = 0; i < array.length; i++)
        for (int j = i + 1; j < array.length; j++)
            if (array[i] > array[j])
                array[i] = array[j];
}
```

E6. Do binárního vyhledávacího stromu reprezentovaného třídou `Node` dopište metodu `void remove(int c)`.

```
class Node {
    Node left, right;
    int contents;

    Node(int c) {
        contents = c;
    }

    void add(int c) {
        if (c < contents)
            if (left == null) left = new Node(c);
            else left.add(c);
        else
            if (right == null) right = new Node(c);
            else right.add(c);
    }
}
```

E7. V databázi jsou novinové články, ke každému z nich je přiřazeno libovolné množství klíčových slov (tagů). Každý článek i každé klíčové slovo jsou reprezentovány řetězcem. Jakou strukturu použijete pro reprezentaci těchto informací tak, aby vložení nového článku a jeho klíčových slov trvalo v průměru lineárně s počtem klíčových slov a aby vypsání všech článků s daným klíčovým slovem trvalo v průměru lineárně s počtem vyhledaných článků? Svoje řešení stručně popište, nezapomeňte zmínit jak bude probíhat vkládání nových článků a vyhledání článků s daným klíčovým slovem.

D1. Na prvním stupni základní školy se děti učí sčítat "na papíře": čísla m a n zapsaná v dekadickém zápise si napíší pod sebe a zleva sčítají odpovídající cifry a případná přetečení z nižších řádů. Jakou má tento algoritmus asymptotickou složitost v závislosti na m a n?

D2. Metoda `put(Object key, Object value)` rozptylovací tabulky `Table` asociuje klíč `key` s hodnotou `value`. V případě, že už nějaká hodnota s daným klíčem asociovaná byla, ji přepíše (tzn. ke stejnému klíči vždy existuje maximálně jedna hodnota). Napište implementaci metody `put`, konkrétní variantu rozptylovací tabulky si zvolte sami.

D3. Quicksort dělí tříděnou posloupnost na tři podposloupnosti: v první jsou prvky menší než pivot, v druhé jsou prvky se stejnou hodnotou jako pivot a ve třetí jsou prvky větší než pivot. Proč by bylo z teoretického pohledu problematické quicksort zjednodušit tak, aby prvky dělil jen na dvě podposloupnosti, prvky menší nebo rovné pivotu a prvky větší než pivot? Předpokládejte, že pivot je volen buď náhodně nebo vždy jako nějaký k-tý prvek (k je konstanta).

D4. Rozptylovací tabulka a a, b-strom řeší velmi podobnou úlohu (mají podobnou sadu operací). a, b-strom oproti rozptylovací tabulce navíc vyžaduje, aby bylo možné pro dva prvky rozhodnout, který z nich je větší. Má a, b-strom proti rozptylovací tabulce nějakou výhodu? Dokáže něco, co rozptylovací tabulka ne (nebo jen velmi neefektivně)?

D5. Popište binární vyhledávací strom, ve kterém lze kromě běžných operací také efektivně (asymptoticky lineárně s výškou stromu) najít k-tý nejmenší prvek. Jak se změní struktura jeho uzlu a jeho operace? Stačí slovní (ale pokud možno exaktní) popis.

C1. Indukcí ukažte proč má a, b-strom vždy stejnou výšku všech větví.

C2. Při řazení vkládáním berete prvky ze vstupního neseřazeného pole zleva doprava a vkládáte je do výstupní setříděné posloupnosti. Vkládání probíhá tak, že nejprve zapíšete prvek na konec posloupnosti a poté ho prohazujete s levým sousedem tak dlouho, dokud není levý soused menší. Odvoďte vzorec, který vám k dané posloupnosti vrátí počet prohození provedených algoritmem řazení vkládáním. Použijte v něm matici $greater_{i,j}$, která má na indexu i, j jedničku právě tehdy, když $i$tý prvek vstupní posloupnosti je větší než $j$tý, jinak nulu. Můžete předpokládat, že vstupní pole neobsahuje dvě stejné hodnoty.

C3. Představte si, že máte metodu `boolean oracle()`, která vám nějakým zázračným způsobem vždy vrátí tu hodnotu, kterou právě potřebujete (tzn. tu, která vás posune ke správnému výsledku). Jak rychle dokážete za použití této metody najít index daného prvku v nesetříděném poli? Předpokládejte, že je zaručeno, že hledaný prvek v poli existuje.

C4. Dokažte: $f \notin \omega(g) \wedge f \notin o(g) \Rightarrow f \in \Theta(g)$.

C5. V Mehlhornově učebnici se neomezené pole zvětšuje na dvojnásobek a zmenšuje při čtvrtinovém naplnění. V čem je čtvrtina natolik významná, proč se nemůže zmenšovat při poloviční naplnění?