Pokud zadání nerozumíte nebo se vám zdá nejednoznačné, zeptejte se. Pište čitelně, nečitelná řešení nebudeme uznávat.

1. (a) Máte tři funkce: f, g a h. Víte, že $f \in O(g)$ a $g \in O(h)$. Z definice dokažte, že $f \in O(h)$.

   (b) Máte tři funkce: f, g a h. Víte, že $f \in \Omega(g)$ a $g \in \Omega(h)$. Z definice dokažte, že $f \in \Omega(h)$.

2. (a) Pro nějaké konstantní číslo k určete asymptotickou časovou složitost v průměrném případě funkce f (předpokládejte, že všechny hodnoty proměnné n jsou stejně pravděpodobné). Svoji odpověď zdůvodněte.

```
int f(int n) { // vždy platí, že 1 <= n <= k
    if (n > 1) return 1;
    int result = 0;
    for (int i = 0; i < k; i++) result += i;
    return result;
}
```

   (b) Pro nějaké konstantní číslo k určete asymptotickou časovou složitost v průměrném případě funkce f (předpokládejte, že všechny hodnoty proměnné n jsou stejně pravděpodobné). Svoji odpověď zdůvodněte.

```
int f(int n) { // vždy platí, že 1 <= n <= k
    if (n > 2) return 1;
    int result = 0;
    for (int i = 0; i < k; i++) result += i;
    return result;
}
```

3. (a) Funkce `min` hledá minimální prvek ze zadaného pole. Na konci každé iterace smyčky mělo platit, že v proměnné result je minimum z části pole mezi indexy 0 a i:

$$\forall j, 0 \leq j \leq i : result \leq array[j] \wedge \exists k, 0 \leq k \leq i : result = array[k].$$

   Najděte ve funkci min chybu a na příkladu ji popište.

```
int min(int[] array) { // délka pole array je nenulová
      int result = 0;
      for (int i = 0; i < array.length; i++)
          if (result > array[i]) result = array[i];
      return result;
}
```

   (b) Funkce `max` hledá maximální prvek ze zadaného pole. Na konci každé iterace smyčky by mělo platit, že v proměnné result je maximum z části pole mezi indexy 0 a i:

$$\forall j, 0 \leq j \leq i : result \geq array[j] \wedge \exists k, 0 \leq k \leq i : result = array[k].$$

   Najděte ve funkci max chybu a na příkladu ji popište.

```
int max(int[] array) { // délka pole array je nenulová
    int result = 0;
    for (int i = 0; i < array.length; i++)
        if (result < array[i]) result = array[i];
    return result;
}
```

4. *Toto zadání bylo pro obě varianty shodné*

   Do třídy List dopište metodu void remove(Object o), která odstraní zadaný prvek ze spojového seznamu. Prvky porovnávejte pomocí equals, dejte si pozor, abyste správně ošetřili všechny krajní případy!

```
class List {
    Node first;

    List() {}
    void add(Object o) {
        first = new Node(o, first);
    }
}

class Node {
    Object contents;
    Node next;

    Node(Object c, Node n) {
        contents = c;
        next = n;
    }
}
```

5. (a) Předpokládejete, že v poli array se každá hodnota vyskytuje maximálně jednou. Jaká musí být minimální velikost k, aby randomizovaná funkce findMin s alespoň padesátiprocentní pravděpodobností vracela minimum? Odvoďte vzorec pro závislost k na n (délce pole) a svoji odpověď zdůvodněte.

```
int findMin(int[] array) {
    int n = array.length;
    int k = ...;
    int min = array[randomInt(0, n)]; // 0 <= randomInt(0, n) < n
    for (int i = 1; i < k; i++) {
        int j = randomInt(0, n);
        if (min > array[j]) min = array[j];
    }
    return min;
}
```

   (b) Předpokládejete, že v poli array se každá hodnota vyskytuje maximálně jednou. Jaká musí být minimální velikost k, aby randomizovaná funkce findMax s alespoň padesátiprocentní pravděpodobností

vracela maximum? Odvoďte vzorec pro závislost k na n (délce pole) a svoji odpověď zdůvodněte.

```
int findMax(int[] array) {
    int n = array.length;
    int k = ...;
    int max = array[randomInt(0, n)]; // 0 <= randomInt(0, n) < n
    for (int i = 1; i < k; i++) {
        int j = randomInt(0, n);
        if (max < array[j]) max = array[j];
    }
    return max;
}
```