Určete a vyznačte tu část kódu, která porušuje daný invariant.

```
int f(int[] a) {
  int r = 1;
  for (int i = 0; i < a.length; i++) {
    r = r + a[i];
    // invariant: v r je soucet prvku pole a na indexech 0..i
  }
  return r;
}
```

Invariant porušuje druhá řádka, měla by zajistit, aby při první iteraci platilo `r == a[0]`, tzn. mělo by na ní být `int r = 0`.

Určete a vyznačte tu část kódu, která porušuje daný invariant.

```
int f(int[] a) { // vzdy plati a.length > 0
  int r = a[0];
  int i = a.length;
  do {
    i--;
    r = r + a[i];
    // invariant: v r je soucet prvku na indexech i..a.length - 1
  } while (i > 0);
}
```

Invariant porušuje druhá řádka, měla by zajistit, aby při první iteraci platilo `r == a[a.length - 1]`, tzn. mělo by na ní být `int r = 0`.

Určete a vyznačte tu část kódu, která porušuje daný invariant.

```
int f(int[] a) {
  int r = 0;
  int i = 0;
  while (i < a.length) {
    r = r * a[i];
    // invariant: v r je soucin prvku pole a na indexech 0..i
    i++;
  }
  return r;
}
```

Invariant porušuje druhá řádka, měla by zajistit, aby při první iteraci platilo `r == a[0]`, tzn. mělo by na ní být `int r = 1`.

Z definice dokažte: $n \in O(n^2)$.
Rozepišme si definici: $n \in O(n^2) \equiv \exists c > 0, n_0 > 0; \forall n > n_0; cn^2 \geq n$. Pro $n_0 = 1$ a $c = 1$ tvrzení $\forall n > n_0; cn^2 \geq n$ určitě platí.

Z definice dokažte: $\sqrt{n} \in O(n)$.
Rozepišme si definici: $\sqrt{n} \in O(n) \equiv \exists c > 0, n_0 > 0; \forall n > n_0; cn \geq \sqrt{n}$. Pro $n_0 = 1$ a $c = 1$ tvrzení $\forall n > n_0; cn \geq \sqrt{n}$ určitě platí.

Z definice dokažte: $3n \in \Theta(n/2)$.
Rozepišme si definici: $3n \in \Theta(n/2) \equiv \exists c_1 > 0, c_2 > 0, n_0 > 0; \forall n > n_0; c_1 \cdot n/2 \leq 3n \leq c_2 \cdot n/2$. Pro $n_0 = 1$, $c_1 = 1$ a $c_2 = 7$ tvrzení $\forall n > n_0; c_1 \cdot n/2 \leq 3n \leq c_2 \cdot n/2$ určitě platí.

Určete asymptotickou složitost funkce `f` v nejhorším případě v závislosti na `n`. Svoji odpověď zdůvodněte.

```
int f(int[] a, int n) {
  if (n == 0) return 1;
  int s = 0;
  for (int i = 0; i < a.length; i++) s += a[i];
  s += f(a, n - 1);
  return s;
}
```

$O($`n`$)$. Funkce `f` se `n`-krát rekurzivně zanoří, práce v těle funkce `f` je konstantní (délku pole `a` považujeme za konstantu).

Určete asymptotickou složitost funkce `f` v nejhorším případě v závislosti na `n`. Svoji odpověď zdůvodněte.

```
int f(int[] a, int n) {
  if (n == 0) return 1;
  int s = f(a, n - 1);
  for (int i = 0; i < a.length; i++) s += a[i];
  return s;
}
```

$O($`n`$)$. Funkce `f` se `n`-krát rekurzivně zanoří, práce v těle funkce `f` je konstantní (délku pole `a` považujeme za konstantu).

Určete asymptotickou složitost funkce `f` v nejhorším případě v závislosti na `a.length`. Svoji odpověď zdůvodněte.

```
boolean f(int[] a, int[] b) { // vzdy plati, ze a.length == b.length
  for (int i = 0; i < a.length; i++)
    if (a[i] != b[b.length - i - 1]) return false;
  return true;
}
```

$O($`a.length`$)$. Tělo for-cyklu proběhne v nejhorším případě `a.length`-krát a práce uvnitř těla je konstantní.

Co vrátí volání `f(a, 0)`? Napište odpověď a demonstrujte ji na následujícím poli: 5, 3, 4, 9, 7, 6, 0, 8, 0.

```
int f(int[] a, int i) {
  if (i == a.length - 1) {
    if (a[i] == 0) return i;
    else return -1;
  }
  if (a[i] == 0) return i;
  else return f(a, i + 1);
}
```

Vrátí pozici nejlevější nuly. Pokud nula v poli není, vrátí -1.

Co udělá volání `f(a, 0, 0)`? Napište odpověď a demonstrujte ji na následujícím poli: 5, 3, 4, 9, 7, 6, 0, 8, 0.

```
void f(int[] a, int i, int j) {
  while ((j < a.length) && (a[j] != 0)) j++;
  if (j >= a.length) return;
  int x = a[j];
  a[j]= a[i];
  a[i] = x;
  f(a, i+1, j+1);
}
```

Přesune nuly v poli doleva.

Co udělá volání `f(a, 0, a.length - 1)`? Napište odpověď a demonstrujte ji na následujícím poli: 5, 3, 4, 9, 7, 6, 0, 8, 0.

```
void f(int[] a, int i, int j) {
  if (i >= j) return;
  int x = a[j];
  a[j]= a[i];
  a[i] = x;
  f(a, i + 1, j - 1);
}
```

Obrátí pořadí prvků v poli.

Jaká je pravděpodobnost, že funkce `f` vrátí minimum z pole `a`. Předpokládejte, že žádné dva prvky v poli `a` nemají stejnou hodnotu. Svoji odpověď zdůvodněte.

```
int f(int[] a) { // vzdy plati a.length > 0
  int r = random(0, a.length - 1); // funkce random vrati nahodne cislo
                                   // mezi 0 a a.length - 1 vcetne
  int min = a[r];
  while (r > 0) {
    r--;
    if (a[r] < min) min = a[r];
  }
  return min;
}
```

Označme si `a.length` $= n$. Jev A_i označuje situaci, ve které je minimum pole `a` na indexu i, $p(A_i) = 1/n$. Jev B_j označuje situaci, ve které má proměnná `r` počáteční hodnotu j, $p(B_j) = 1/n$. Pokud je počáteční hodnota `r` napravo od minima (tzn. nastaly jevy A_i a B_j, kde $i \leq j$), funkce `f` minimum vrátí. Pravděpodobnost takového jevu je

$$p = \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} p(A_i) \cdot p(B_j) = \frac{(n+1)n}{2} \cdot \frac{1}{n} \cdot \frac{1}{n} = \frac{n+1}{2n}$$

Jaká je pravděpodobnost, že funkce `f` vrátí `true`? Předpokládejte, že pole `a` i `b` obsahují náhodnou permutaci prvků $1 \ldots n$ (a tedy mají shodnou délku). Svoji odpověď zdůvodněte.

```
boolean f(int[] a, int[] b) {
  int r1 = random(0, a.length - 1); // funkce random vrati nahodne cislo
                                    // mezi 0 a a.length - 1 vcetne
  int r2 = random(0, b.length - 1);
  return a[r1] == b[r2];
}
```

Označme si `a.length` $= n$. Pravděpodobnost, že na indexu `r1` je hodnota k, je $1/n$. Pravděpodobnost, že na indexu `r2` je hodnota k, je $1/n$. Oba jevy jsou na sobě nezávislé, takže pravděpodobnost, že `a[r1] == b[r2]`, je

$$\sum_{k=1}^{n} \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n}$$

Jaká je pravděpodobnost, že funkce `f` vrátí `true`? Předpokládejte, že pole `a` i `b` obsahují náhodnou permutaci prvků $1 \ldots n$ (a tedy mají shodnou délku). Svoji odpověď zdůvodněte.

```
boolean f(int[] a, int[] b) {
  int r1 = random(0, a.length - 1); // funkce random vrati nahodne cislo
                                    // mezi 0 a a.length - 1 vcetne
  int r2 = random(0, b.length - 1);
  return a[r1] <= b[r2];
}
```

Označme si `a.length` $= n$. Pravděpodobnost, že na indexu `r1` je hodnota k, je $1/n$. Pravděpodobnost, že na indexu `r2` je hodnota l, je $1/n$. Oba jevy jsou na sobě nezávislé, takže pravděpodobnost, že `a[r1] <= b[r2]`, je

$$\sum_{k=1}^{n} \sum_{l=k}^{n} \frac{1}{n} \cdot \frac{1}{n} = \frac{(n+1)n}{2} \cdot \frac{1}{n} \cdot \frac{1}{n} = \frac{n+1}{2n}$$