1. Za abstraktni podatkovni tip Int z definiranimi operacijami **succ, pred, add, neg in sub** pokažite, da je: **sub(neg(n), m) = neg(add(n, m))**.

Vemo:

- succ(0) = 1,
- succ(succ(succ(…(0)…))) (n-krat) lahko zapišemo kot $succ^n(0) = n$,
- neg(n) = $succ^{-n}(0)$,
- pred(0) = -1,
- pred(pred(pred(…(0)…))) (n-krat) lahko zapišemo kot $pred^n(0) = neg(succ^n(0))$

Dokazujemo resničnost: **sub(neg(n), m) = neg(add(n, m))**

Leva stran:

n … $succ^n(0)$
m … $succ^m(0)$

1. sub(neg(n), m)
2. sub(neg($succ^n(0)$), $succ^m(0)$)
3. sub($succ^{-n}(0)$, $succ^m(0)$)
4. $succ^{-(n-m)}(0)$
5. $\underline{pred^{n+m}(0)}$

Desna stran:

n … $succ^n(0)$
m … $succ^m(0)$

1. neg(add(n, m))
2. neg(add($succ^n(0)$, $succ^m(0)$)
3. neg($succ^{n+m}(0)$)
4. $\underline{pred^{n+m}(0)}$

Sklep: enakost velja.

2. Napišite javansko implementacijo metode **obrni(Stack s, int n, int m)**, ki obrne m elementov sklada s od mesta n dalje. Rezultat je spremenjen izhodiščni sklad, pri reševanju pa lahko uporabljate pomožne sklade. Za izvedbo sklada uporabite razred Stack iz Collection Framework-a. Rešitev naj vsebuje samo operacije nad skladi.

```
static void obrni(Stack<Integer> s, int n, int m) {
    Stack<Integer> sklad1 = new Stack<>();
    Stack<Integer> sklad2 = new Stack<>();
    Stack<Integer> sklad3 = new Stack<>();
    for (int i = 0; i < n; i++) {
        sklad1.push(s.pop());
    }
    for (int i = 0; i < m; i++) {
        sklad2.push(s.pop());
    }
    for (int i = 0; i < m; i++) {
        sklad3.push(sklad2.pop());
    }
    for (int i = 0; i < m; i++) {
        s.push(sklad3.pop());
    }
    for (int i = 0; i < n; i++) {
        s.push(sklad1.pop());
    }
}
```