Exercise 1

Consider the Java class `IntegerInterval` that represents an interval of integer values.

```java
class IntegerInterval {
    int getLowerBound() { ... }
    int getUpperBound() { ... }
    void doSomething (int i) { ... }
}
```

The methods of the class `IntegerInterval` have the following specifications:

- `getLowerBound()`: @pre: `true`; @post: `0 <= getLowerBound() < getUpperBound()`
- `getUpperBound()`: @pre: `true`; @post: `0 <= getLowerBound() < getUpperBound()`
- `doSomething(int i)`: @pre: `getLowerBound() <= i < getUpperBound()`; @post: `true`;

Additionally, consider the class `NegativeIntegerInterval` that extends `IntegerInterval` as follows.

```java
class NegativeIntegerInterval extends IntegerInterval {
    void doSomething (int i) {
        super.doSomething (-i);
    }
}
```

The method `doSomething` in the class `NegativeIntegerInterval` has the following specification:

- `doSomething(int i)`: @pre: `this.getLowerBound() <= -i < this.getUpperBound()`; @post: `true`

Consider the class `Run` that uses the `NegativeIntegerInterval` class as follows.
class Run {

    public static void main (String[] a) {

        IntegerInterval c = new NegativeIntegerInterval();

        c.doSomething(-42);
        c.doSomething(42);

    }

}

Analyze the code and identify whether contract violations may occur during run-time.

Exercise 2

Prove the partial correctness of the programs specified by the following Hoare triples.

(i) @pre = { x >= 10, y >= 0 }

    y = y + x;

    @post = { x >= 0, y >= 5 }

(ii) @pre = { true }

    if (a > b) {
        m = a;
    } else {
        m = b;
    }

    @post = { m == max (a, b) }

(iii) @pre = { n >= 0 }

    int sum = 0;
    int i = 0;
    while (i < n) {
        i = i + 1;
        sum = sum + i;
    }

    @post = { sum == n * (n + 1)/2 }

    Hint: Prove first that INV ≡ (sum + \sum_{j=i+1}^{n} j == n(n+1)/2) \land i \leq n is a loop invariant.
Exercise 3

Identify (i) the basic paths in the following program, and (ii) compute the verification conditions VCs for the basic paths. Are the VCs valid?

@pre = { true }

if (a > b) {
    m = a;
} else {
    m = b;
}

@post = { m == max (a, b) }