# Methodologies for Software Processes

Seminar 4

## Assignment 4

- Please complete the following tasks until Seminar 5 or Seminar 6.
- The Assignment 4 must be presented at the Seminar 5/Seminar 6 (all group members must be in the class).

- Write a recursive method `sum` that yields the sum of the first n natural numbers.
- Provide a suitable specification.
- Check whether your specification is strong enough by verifying the client code below.

```
method main() {
  var r: Int
  r := sum(10)
  assert r == 55
}
```

- Use Viper to prove that McCarthy's 91 function (right) terminates.

```
method M(n: Int) returns (r: Int)
  requires n >= 0
  ensures 100 < n  ==> r == n - 10
  ensures n <= 100 ==> r == 91
{
  if (n > 100) {
    r := n - 10
  } else {
    r := M(n + 11)
    r := M(r)
  }
}
```

- The file *`03-trees.vpr`* axiomatizes binary trees with integer values stored in leafs.
- Extend the `Tree` domain by a function `size` that takes a Tree and returns the number of leafs in the tree.
- Extend the `Tree` domain by a function `sum` that takes a Tree and returns the sum of all values stored in the tree.
- Test your domain against the following client (also found in the file but commented out)

```
method client() {
    var t: Tree
    t := node(
            node(
              leaf(3),
              leaf(17)
            ),
            leaf(22)
         )

    assert sum(t) == 42
    assert size(t) == 3
}
```

Define a function fib(n) that yields the nth Fibonacci number.

```
fib(0)   = 0
fib(1)   = 1
fib(n+2) = fib(n+1) + fib(n)
```

Provide a suitable precondition.

Verify that the method on the right computes the nth Fibonacci number.

Hint: You can use the skeleton `07-fib.vpr`

```
method iter_fib(n: Int) returns (res: Int)
  requires 0 <= n
  ensures  …
{
  res := 0
  var i: Int := 0
  var next: Int := 1

  while (i < n)
    invariant …
  {
    var t: Int := res
    res := next
    next := t + next
    i := i + 1
  }
}
```

- Add a function `size(t: Tree): Int` to the skeleton `10-trees.vpr` that counts the number of leafs in the tree t.

- Add a postcondition such that the client in the code skeleton verifies.

```
method client() {
  var t: Tree
  t := node(node(leaf(3), leaf(17)), leaf(22))
  assert size(t) >= 0
}
```