University of Tartu *Fall 2014*
Department of Computer Science *Vesal Vojdani*

Formal Methods in Software Engineering

Exercise sheet 1 (preparation)

Exercise 1: *Factorial example from the lecture* *No Points*

This first "exercise" is not really an exercise; it just shows how to write down the full Tableaux proof on paper. This is the factorial function in original code:

```
int fact(int x) {
    y = 1;
    z = 0;
    while (z != x) {
        z = z + 1;
        y = y * z;
    }
    return y;
}
```

There is no return statement in the While language. We show that when y is returned the equality $y = x!$ holds. Last lecture, we stepped through a few iterations of the loop and guessed that the invariant should be $y = z!$, which we can use to attempt a proof:

$$
\begin{array}{l}
\quad (\!|\, true \,|\!) \\
\quad (\!|\, 1 = 0! \,|\!) \\
y = 1\ ; \\
\quad (\!|\, y = 0! \,|\!) \\
z = 0\ ; \\
\quad (\!|\, y = z! \,|\!) \\
\textsf{while}\ z \neq x\ \textsf{do}\ \{ \\
\qquad\quad (\!|\, (y = z!) \wedge (z \neq x) \,|\!) \\
\qquad\quad (\!|\, y \cdot (z+1) = (z+1)! \,|\!) \\
\qquad z = z + 1\ ; \\
\qquad\quad (\!|\, y \cdot z = z! \,|\!) \\
\qquad y = y \cdot z \\
\qquad\quad (\!|\, y = z! \,|\!) \\
\} \\
\quad (\!|\, (y = z!) \wedge (z = x) \,|\!) \\
\quad (\!|\, x = y! \,|\!)
\end{array}
$$

Exercise 2: *What is the invariant?* *No Points*

Consider now the following program:

$$
\begin{array}{l}
\quad (\!|\, \mathit{true} \,|\!) \\
x = 2 \cdot y \; ; \\
z = 0 \; ; \\
\textsf{while } z \neq x \textsf{ do } \{ \\
\qquad z = z + 1 \; ; \\
\qquad x = x - 1 \\
\} \\
\quad (\!|\, z = y \,|\!)
\end{array}
$$

How do we infer an invariant for this program?

Exercise 3: *Multiplication Counting Up* *No Points*

Show that this program computes $z = x \cdot y$:

```
a = 0;
z = 0;
while (a != y) {
     z = z + x;
     a = a + 1;
}
```

Exercise 4: *Multiplication Counting Down* *No Points*

Show that this program computes $z = x \cdot y$:

```
z = 0;
while (y != 0) {
     z = z + x;
     y = y - 1;
}
```

Note that y changes during computation! How can we express this in pre- and post-conditions?