Program Verification Prof. Dr. H. Peter Gumm
Summer 2023

Exercise sheet 11

Deadline: July 08, 8:00 p.m.

Please submit only a Dafny-file **ex11_*your_name*.dfy**.

Problem 1 (3 points)**.** The factorial function can be defined as follows:

```
ghost function factorial(n:nat):nat
{ if n == 0 then 1 else n * factorial(n-1)}
```

We can implement it as a recursive **method**, whose specification will be automatically verified by Dafny:

```
method fact(n:int) returns (f:int)
requires n >= 0
ensures f == factorial(n)
{
  if n == 0 { f := 1;}
  else {
    f := fact(n-1);
    f := f*n;
  }
}
```

Keeping the specification and general structure of this method, your task is to translate the body of *fact* into the language of *Boogie* - that is you are *not allowed* to use any method calls, rather you are supposed to replace the recursive call to *fact* using the means supplied by *Boogie*, as there are: *var* declaration, *assert* and *assume*. In the end your new method *fact* should verify again.

Problem 2 (3 points)**.** Starting with the following definition of the *fibonacci*-function

```
function fib(n:nat):nat
{ if n < 2 then n else fib(n-2) + fib(n-1)}
```

use Dafny to state and prove a lemma *fibExtends* stating that for all $n \geq 5$ we have $n \leq fib(n)$. Since the proof is almost trivial, turn induction off by adding the annotation `{:induction false}`, as in

```
lemma {:induction false} fibExtends(n:nat)
```

Problem 3 (3 pts)**.** Given the definition of *divisibility*, as in

```
ghost predicate divides(d:int,n:int)
  requires d ≠ 0
{ ∃ k:int :: k*d == n}
```

State and prove a lemma, expressing that if d divides m and n, then d also divides $m + n$ and $m - n$.

Problem 4 (4 pts)**.** The gcd-function can be specified as follows:

```
ghost function gcd(x:nat,y:nat):nat
  decreases x+y,y
{
  if x == 0 || y == 0 then x+y
  else if x > y then gcd(x-y,y)
  else gcd(x,y-x)
}
```

An important mathematical fact, known as *Bézout's lemma,* states that for any numbers x, y there exist integers a and b such that $gcd(x, y) = a * x + b * y$.

To find a and b you can modify the same algorithm which you use to calculate gcd, except that instead of starting with x and y you start with equations "$x = \mathbf{1} * x + \mathbf{0} * y$" and "$y = \mathbf{0} * x + \mathbf{1} * y$". Then instead of manipulating x and y, extend these same manipulations to the coefficients of the corresponding equations.

For instance, when calculating $gcd(70, 48)$, you would normally calculate

$$gcd(70, 48) = gcd(32, 48) = gcd(48, 32) = gcd(16, 32) = gcd(32, 16) = gcd(16, 16) = 16.$$

To find the required a and b, start with the equations $70 = \mathbf{1} * 70 + \mathbf{0} * 48$ and $48 = \mathbf{0} * 70 + \mathbf{1} * 48$ and manipulate these equations the same way as you manipulate the numbers x, y, resulting in the sequence:

$$\begin{aligned}
70 =& \mathbf{1} * 70 + \mathbf{0} * 48 \\
48 =& \mathbf{0} * 70 + \mathbf{1} * 48 \\
32 =& \mathbf{1} * 70 + (\mathbf{-1}) * 48 \\
16 =& (\mathbf{-1}) * 70 + \mathbf{2} * 48 \\
16 =& \mathbf{2} * 70 + (\mathbf{-3}) * 48 \\
0 =& -3 * 70 + 5 * 48
\end{aligned}$$

As this example shows, a and b are *not* uniquely determined.

Your task is to modify the gcd calculation to yield a module `Bezout(x:nat,y:nat)returns (gcd,a,b)`. Specify, implement and verify this method.

Hint: It is probably easiest, if you first rewrite the above function gcd as a method `gcd(x:nat,y:nat) returns(ggt:nat)`, using a while-loop instead of recursive calls. Then augment it with extra variables, say a, b, a', b' to manipulate the coefficients of x and y in the equations.

Problem 5 (3 pts)**.** Example 5.6 in the book defines a multiplication by means of repeated addition and subtraction:

```
function Mult(x:nat, y:nat):nat
{  if y == 0 then 0 else x + Mult(x,y-1) }
```

State and prove the following lemmas, using Dafny's `calc` feature as explained in chapter 5.4 in the book:

(a). $Mult$ is associative, i.e. $Mult(x, Mult(y, z)) == Mult(Mult(x, y), z)$

(b). $Mult(x, y) == x * y$. Since Dafny will very likely prove this by herself, turn off automatic induction!