Lab session 8: Programming Fundamentals

Lab session 8 consist of seven exercises, and is very similar to the structure of the 2/3rd exam. The problems 1 and 5 are worth 20 points, all other problems are worth 10 points. You get one point for free, making a total of 100 points (=grade 10). All exercises are about proofs with Dafny. Note that it is very easy to fake a correct submission in Themis (for example, make **false** a precondition or **true** a postcondition of a method), but of course that is not allowed. As a result, your grade in Themis is an upperbound for your real grade, and the teaching assistants will check manually all accepted submissions for 'cheating'/misleading Themis. If such abuse is detected, then no points for the corresponding exercise will be awarded.

Problem 1 (20 points): MCQ (Multiple Choice Questions

For each of the following annotations determine which choice fits on the empty line (.....). The variables `x`, `y`, and `z` are of type `int`. Note that `A` and `B` (capital letters!) are specification constants (so not program variables).

```
1. // 10 < 2*x + 6*y < 20
   .....
   // 10 < x < 15

   (a) x := 2*x + 6*y - 5;
   (b) x := 2*x + 6*y + 5;
   (c) x := x + 3*y + 5;

2. // x == A + B && y == A - B
   .....
   // x == A && y == B

   (a) x := x/2; y := x - y;
   (b) y := (x - y)/2; x := x - y;
   (c) y := x - y; x := x/2;

3. // x == A && y == B
   .....
   // // x - 2*B == A

   (a) x := x + y; y := x + y;
   (b) y := x + y; x := x + y;
   (c) x := x + y; x := x + y;
```

```
4. .....
   y := 2*(x+y); x := 2*(x+y);
   // x == 6*A + 4*B && y == 2*A + 2*B


   (a) // x + y == 3*A + 2*B && 3*x + y == A + B
   (b) // x == A && y == B
   (c) // x + y == 3*A + 4*B && 2*(x + y) == 2*A + 2*B

5. // x == A && y == B
   z := x; x := y; y := z;
   .....

   (a) // x == B && y == A && z == A
   (b) // x == A && y == A && z == B
   (c) // x == A && y == B && z == A

6. // y == A && x == z == B
   z := x - y; x := x + y + z; y := z - y;
   .....

   (a) // x == 3*B && y == B - A && z == A + 2*B
   (b) // x == 2*B && y == B - 2*A && z == B - A
   (c) // x == A + 2*B && y == B - A && z == B - A
```

Problem 2 (10 points)

From Themis you can download the file `problem2.dfy`, which contains:

```
method problem2(p:int, q:int, ghost X:int, ghost Y:int)
   returns (r:int, s:int)
requires p == 2*X + Y && q == X + 3
ensures r == X && s == Y
{
  // X and Y are specification constants and are not allowed to
  // appear in the body of this method.
  //
  // implement yourself
  ...........
}
```

Fill in the dots such that the Dafny verifier accepts the method.

Problem 3 (10 points)

From Themis you can download the file `problem3.dfy`, which contains:

```
method problem3(m:int, X:int) returns (r:int)
requires X >= 0 && (2*m == 1 - X || m == X + 3)
ensures r == X
{
  // X is a specification constant and is not allowed to
  //  appear in the body of this method.
  //
  // implement yourself
  ...........
}
```

Fill in the dots such that the Dafny verifier accepts the method.

Problem 4 (10 points)

From Themis you can download the file `problem4.dfy`, which contains:

```
method problem4(a: nat, b: nat)
{
  var i,j: int;
  i,j := a,b;
  while i > 0 && j > 0
  decreases .......      // choose a suitable variant function
  {
    if i < j {
      i,j := j,i;
    } else {
      i := i - 1;
    }
  }
}
```

Find a suitable variant function such that the Dafny verifier is able to prove termination.

Problem 5 (20 points)

From Themis you can download the file `problem5.dfy`, which contains:

```
ghost function f(n: int): int {
  // you are not allowed to remove 'ghost', so an asignment
  // like x := f(n) is not allowed.
  if n < 0 then 0 else 3*f(n-5) + n
}

method problem5(n:nat) returns (x: int)
ensures x == f(n)
{
  // implement yourself. You are required to give an invariant!
}
```

Fill in the dots such that the Dafny verifier accepts the method.

Problem 6 (10 points)

From Themis you can download the file `problem6.dfy`, which contains:

```
ghost function f(n: int): int {
  if n <= 0 then 1 else n + f(n-1)*f(n-2)
}

ghost function fSum(n: nat): int {
 // Implement the body of this function. It should return Sum(i: 0<=i<n: f(i))
  ...........
}

method problem6(n:nat) returns (a: int)
ensures a == fSum(n)
{
  var k, x, y;
  a,k,x,y := ...........;  // initialize yourself
  while k < n
    invariant 0<=k<=n && x==f(k) && y==f(k+1) && a == fSum(k)
  {
    k := k + 1;
    // complete the rest of this method
    a := ......;
    x,y := ......;
  }
}
```

Fill in the dots. The function `fSum(n)` should return $\Sigma(i:\ 0 \leq i < \mathtt{n}:\ \mathtt{f}(i))$.

Problem 7: Polynomial evaluation (10 points)

A polynomial $\sum_{i=0}^{n-1} a_i \cdot x^i = a_0 \cdot x^0 + a_1 \cdot x^1 + \ldots + a_{n-1}x^{n-1}$ can be represented by an array `a` where $a_i =$ `a[i]`. From Themis you can download the file `polynomial.dfy`. It contains the following code fragment:

```
ghost function exp(x: int, e: nat): int
{ // returns x raised to the power e
  if e == 0 then 1 else x*exp(x, e-1)
}

ghost function polynomial(x: int, n: nat, a: array<int>): int
requires n <= a.Length
reads a
{ // returns Sum(i in [0..n):: a[i]*x^i)
  if n == 0 then 0 else a[n-1]*exp(x, n-1) + polynomial(x, n-1, a)
}

method Polynomial(x: int, a: array<int>) returns (p : int)
  ensures p == polynomial(x, a.Length, a)
{
  ...... // implement yourself
}
```

Implement the method `Polynomial` such that the call `Polynomial(x,a)` performs at most `a.Length` computation steps.