Suggested Solutions to HW#5 Problems

1. (40 points) Consider the structure $\mathcal{Z} = (\mathbb{Z}, \{+, \times, 0, 1, <\})$, i.e., the set of integers with the usual arithmetic functions ($+$ and $\times$), constants (0 and 1), and predicates ($<$); "=" is implicitly assumed to be a binary predicate.

   (a) Let $divides(a, b)$, or alternatively $a \mid b$, denote that a divides b. Write a formula defining $divides(a, b)$ (so, your formula will have a and b as free variables).

   Solution. $divides(a, b) \stackrel{\Delta}{=} \exists q(b = a \times q)$. □

   (b) Let $isGCD(a, b, c)$ denote that c is the greatest common divisor of a and b. Write a formula defining $isGCD(a, b, c)$.

   Solution. $isGCD(a, b, c) \stackrel{\Delta}{=} divides(c, a) \wedge divides(c, b) \wedge \forall d(divides(d, a) \wedge divides(d, b) \rightarrow divides(d, c))$ □

2. (20 %) The following C function `originalEuclid` implements the original Euclidean algorithm. Please give a suitable function contract, namely the pre and post-conditions, for `originalEuclid`, using either ACSL or the conventional logic notation. Use `\result` to denote the value returned by a function. Let us assume that, for this problem, the type `int` is the same as the set $\mathbb{Z}$ of integers. And, the formulae you will write are intended for the semantic structure $\mathcal{Z} = (\mathbb{Z}, \{+, -, \times, 0, 1, <\})$, same as in the previous problem; you may reuse definitions from there.

```
int originalEuclid (int m, int n)
{ int x,y,tmp;

  x = m;
  y = n;
  while (x!=y) {
    if (x < y) {
      tmp = x;
      x = y;
      y = tmp;
    }
    x = x - y;
  }
  return x;
}
```

Solution. Precondition: $(0 < m) \wedge (0 < n)$.

Postcondition: $(0 < \texttt{\textbackslash result}) \wedge isGCD(m, n, \texttt{\textbackslash result})$, where $isGCD$ is as defined in the previous problem. □

3. (40 %) Please examine the following C function `sumofMM` and give a suitable function contract, namely the pre and post-conditions, for `sumofMM`, using either ACSL or the conventional logic notation. Use `\result` to denote the value returned by a function. Be careful that the function has made an implicit assumption about the size of the input array (which means that you should put it in the precondition). You may omit the condition concerning proper memory allocation for the input array and focus on what the function does. Let us assume that, for this problem, the type `int` is the same as the set $\mathbb{Z}$ of integers. The formulae you will write are intended for the semantic structure $\mathcal{Z} = (\mathbb{Z}, \{a[\,], +, -, *, 0, 1, 2, <\})$; "=" is implicitly assumed to be a binary predicate as usual.

```
int sumofMM (int* a, int n)
{ int min, max, i;

  // Initialize min and max.

  if (a[0] < a[1]) {
    min = a[0];
    max = a[1];
  }
  else {
    min = a[1];
    max = a[0];
  }

  // Divide the rest into pairs.
  // Compare the smaller with min and the larger with max.

  for (i=2; i<n; i=i+2)
    if (a[i] < a[i+1]) {
      if (a[i] < min)
        min = a[i];
      if (a[i+1] > max)
        max = a[i+1];
    }
    else {
      if (a[i+1] < min)
        min = a[i+1];
```

```
      if (a[i] > max)
        max = a[i];
    }

  return min+max;
}
```

Solution. Precondition: $(2 \leq n) \wedge \exists m(2 \times m = n)$. (Function `sumofMM` assumes that n is a positive even integer.)

Postcondition: $\exists i(\exists j((0 \leq i < n) \wedge \forall k((0 \leq k < n) \rightarrow (a[i] \leq a[k])) \wedge (0 \leq j < n) \wedge \forall k((0 \leq k < n) \rightarrow (a[i] \leq a[k])) \wedge (a[i] + a[j] = \texttt{\textbackslash result})))$. (The returned value of `sumofMM` is the sum of the minimum and the maximum of the numbers in the input array.) □