# Problem Sheet 4

## Guarded Command Language

**Q1.** Using the guarded command language described above write programs to solve each of the following problems:

- Write a program to read any valid date this century and generate its successor date.
- Read the lengths of three lines, determine if they form a triangle and, if so, print one of scalene, isosceles or equilateral.
- Given an integer x determine if it is a prime number.
- Determine the integer square root of $x$, i.e. find $a$ such that $a^2 \leq x < (a+1)^2$.
- Initialise an array of size N to random values in the range 0..100 and compute the sum of the elements in the array.

**Q2.** Read the following programs carefully and write assertions describing the current states of the variables at the points indicated in the code.

```
|[
  var x: int;
  var b: bool;
  x := rand(1000);
  if x mod 20 = 0 →
        b := true
   [] x mod 20 ≠ 0 →
        b := false
   fi;
   { .. }
]|
```

```
|[ con N : int;
  { N ≥ 1 }
  var t : int;
  t := 1;
  { .. }
  do t < N →
     t := t + 1
  od
  { .. }
 ]|
```

```
|[ con N : int; { N ≥ 0 }
  con x : int; { 0 ≤ x < N }
  var n : int;
  var freq : int;
  var f : array[0..N) of int;
  var fnd : bool;
  n : = 0;
  do n < N →
     f.n := rand(N);
     n := n + 1
  od
  { .. }
  n : = 0; fnd:= false;
  { .. }
  do n < N  ∧ ¬fnd→
      if f.n = x →
        fnd := true
      [] f.n ≠ x →
         skip
      fi;
       n := n + 1
  od
  { .. }
]|
```

**Q3.** Given f[0..N) of integer, sort the elements into ascending order using either Bubblesort, Selection sort or insertion sort. Try to write assertions that describe the state of the array at the end of each loop.

**Q4:** Generate a "magic square" of size N, where N an odd number.

| | | | | |
|---|---|---|---|---|
| 17 | 24 | 1 | 8 | 15 |
| 23 | 5 | 7 | 14 | 16 |
| 4 | 6 | 13 | 20 | 22 |
| 10 | 12 | 19 | 21 | 3 |
| 11 | 18 | 25 | 2 | 9 |

Algorithm: Distribute the numbers 1..N*N starting in middle of top row proceeding north east and wrapping around when necessary or dropping down one row when the number is divisible by N.

**Q5:** A can containing coffee beans is given. Some of the coffee beans are black and the rest white. We play the following game:

Repeatedly pick two beans at random from the can. If the two beans have the same colour then discard them and put in one black bean ( a large supply of extra black beans is available). If the two beans have different colours then replace the white one and discard the black. This process is repeated until only one bean remains in the can. What colour is this bean?

**Writing Specifications**

1. Write specifications for each of the following problems:

    - x equals the quotient on dividing a by b, b not equal to 0;
    - x equals the remainder on dividing a by b, a greater than or equal to 0, b not equal to 0;
    - q, r equal the quotient and remainder on dividing a by b, b not equal to 0;
    - swap the values of two variables;
    - x, y are sorted into ascending order;
    - y is a leap year in the Twenty First Century;
    - b is equivalent to x is a positive whole number;
    - k is equivalent to x is an even number in the range 1..1000;
    - k is equivalent to y is a prime number;
    - k is the integer square root of n, n ≥ 0;
    - b is equivalent to l1, l2, l3 form a tringle;
    - given l1, l2, l3 forming a triangle b is equivalent to l1, l2, l3 form an isosceles triangle.

2. Write specification for each of the following problems involving sequences. In each case let f[0..N), N ≥ 0, be an integer array.

    - b is equivalent to all the elements of f are positive
    - array t is a copy of f
    - sum equals the sum of the elements in f
    - t = the product of the elements in f
    - s = the sum of all the odd elements in f
    - s1 = the sum of all the odd elements and s2 = the sum of all the even elements in f
    - t = the frequency of positive values in f
    - t = the frequency of elements in f in the range 1..100
    - b is equivalent to all the elements in f are sorted into ascending order
    - b is equivalent to all the elements of f are unique
    - max is the largest value in f
    - j = the index of the smallest value in f
    - t is a copy of f and t is sorted
    - t is a copy of f and t[0..i) is sorted and j = the index of the smallest element in t[i..N)