◊ **Best before:** term test 1.

1. Do exercise 4 on page 71 of the course notes (about number of occurrences of an item in an array).

2. This question concerns programs that find the largest integer power of 2 which does not exceed m/n. I.e., we want to compute $2^{\lfloor \log_2(m/n) \rfloor}$.

   (a) Prove this iterative program to be correct with respect to its precondition/postcondition pair — prove both partial correctness and termination.

   ```
   ▷ Precondition: m, n ∈ ℕ, 1 ≤ n ≤ m.
   ▷ Postcondition: Return x such that x = 2^t for some t ∈ ℕ and nx ≤ m < 2nx.
   POW2ITER(m, n)
   x = 1;   y = 2 * n
   while y ≤ m:
       y = 2 * y;   x = 2 * x
   return x
   ```

   (b) Prove this recursive program to be correct with respect to its precondition/postcondition pair.

   ```
   ▷ Precondition: m, n ∈ ℕ, 1 ≤ n ≤ m.
   ▷ Postcondition: Return x such that x = 2^t for some t ∈ ℕ and nx ≤ m < 2nx.
   POW2REC(m, n)
   if m < 2 * n:
       result = 1
   else:
       result = POW2REC(m, n * 2) * 2
   return result
   ```

3. (a) Here is a recursive version of the program on page 59 of the course notes — the specification is the same.

   ```
   ▷ Precondition: x, y ∈ ℕ.
   ▷ Postcondition: True.
   PAGE59(x, y)
   if x == 0 and y == 0:
       return
   elif x ≠ 0:
       PAGE59(x − 1, y)
   else:
       PAGE59(16, y − 1)
   ```

   Prove that PAGE59 is correct with respect to its specification.

   (b) Do exercises 8 and 9 on pages 72-73 of the course notes (about proving program termination).

   (c) Similar to part (a), write recursive versions of the programs from part (b), and provide proofs of correctness for them.
   Advice: For some programs, you may find it convenient to modify the precondition slightly.

4. (a) The following recursive program performs *long division* in base 2 and returns both quotient and remainder. Prove that it is correct with respect to its precondition/postcondition pair.

```
▷ Precondition: x, y ∈ ℕ, y > 0.
▷ Postcondition: Return ⟨x div y, x mod y⟩.
D(x, y)
    if x == 0:
        return ⟨0, 0⟩
    else:
        ⟨q, r⟩ = D(x div 2, y)
        if 2 * r + (x mod 2) < y:
            return ⟨2 * q, 2 * r + (x mod 2)⟩
        else:
            return ⟨2 * q + 1, 2 * r + (x mod 2) − y⟩
```

(b) Write an iterative version of the program $D(x, y)$ from part (a), then prove it to be correct with respect to the same precondition/postcondition pair.

Advice: You may find it helpful to use the Pow2Iter program from question 2.

5. Consider the problem of finding the largest index of a largest element in a slice of a list.

(a) In the following recursive program, A is a list of integers and all other variables are integers. Prove it to be correct with respect to its precondition/postcondition pair.

```
▷ Precondition: 0 ≤ f < l ≤ len(A).
▷ Postcondition: Return an integer u such that
▷   (i) f ≤ u < l,
▷   (ii) A[u] is the largest value in A[f : l], and
▷   (iii) A[u] is greater than every integer in A[u + 1 : l].
Max(A, f, l)
    if f + 1 == l:
        return f
    else:
        m = (f + l) div 2
        x = Max(A, f, m)
        y = Max(A, m, l)
        if A[x] > A[y]:
            return x
        else:
            return y
```

(b) Write an iterative version of the program $\textsc{Max}(A, f, l)$ from part (a), then prove it to be correct with respect to the same precondition/postcondition pair.

6. For this question we consider numbers represented in base 3 and whether they are divisible by 4.

Let $\Sigma_3 = \{0, 1, 2\}$. For a string $x \in \Sigma_3^*$ (we call x a *ternary* string), we define $V(x)$ to be the number represented in base 3 by x. More formally, if $x = x[0]\, x[1] \cdots x[n-1]$, where $n > 0$ and each $x[i] \in \Sigma_3$, then $V(x) = \sum_{i=0}^{n-1} \left(x[n-1-i] \cdot 3^i\right)$.[1] For example, $V(021101) = 199$.

[1] Note that a reference to an element of x can mean either a symbol from Σ_3 or the numeric value of that symbol. E.g., if x is the string 10021, then $x[3]$ is both the symbol 2 and the number 2. You should be able to tell which by its context.

(a) Prove that the following program is correct with respect to its precondition/postcondition pair.

▷ Precondition: $x \in \Sigma_3^*$ and $x \neq \epsilon$ (i.e., x is a nonempty ternary string).
▷ Postcondition: Return **True** if $V(x) \bmod 4 = 0$; Otherwise return **False**.

```
BASE3DIVBY4(x)
    t = x[0];   i = 1
    while i < len(x):
        if t == 0:
            t = x[i]
        else:
            t = 4 - t + x[i]
            if t >= 4:   t = t - 4
        i = i + 1
    return (t == 0)
```

(b) Write a recursive version of the program BASE3DIVBY4(x) from part (a), then prove it to be correct with respect to the same precondition/postcondition pair.

7. *Extra — a programming question!*

The following definition may look familiar.

We say that a nonempty list t of distinct integers has the *bifurcate property* iff this condition holds.

Bifurcate property: For any nonempty slice $t[p : q]$, there is an integer k such that
- $p < k \leq q$,
- every integer in $t[p + 1 : k]$ is less than $t[p]$,
- every integer in $t[k : q]$ is greater than $t[p]$.

Write a program that takes a nonempty list t of distinct integers as input and returns True or False depending on whether t has the bifurcate property, then prove that it is correct.

- Make your program as efficient as possible.
  There are $O(n^2)$ slices in a list of length n. It takes $O(n)$ time to check a slice for the property. So it seems the total complexity of this program is $O(n^3)$. Can you do better?
- You should consider breaking the program into smaller pieces — *modular design!*
  It could simplify your proof.
- Try using only recursion, only iteration or perhaps a combination of both.