Question 1 **Structural Induction over Lists** [2 + 3 + 3 + 11 + 6 Credits]

Consider the functions `filter`, `revAppend` and `length` defined on lists of an arbitrary type `a`

```
filter :: (a -> Bool) -> [a] -> [a]
filter p []   = []                      -- F1
filter p (x:xs)
  | p x       = x:(filter p xs)         -- F2
  | otherwise = filter p xs             -- F3

revAppend :: [a] -> [a] -> [a]
revAppend   []  ys  = ys                -- RA1
revAppend (x:xs) ys = revAppend xs (x:ys) -- RA2

length :: [a] -> Int
length   []   = 0                       -- L1
length (x:xs) = 1 + length xs           -- L2
```

The goal of this exercise is to show that

```
length (filter p (revAppend xs ys)) = length (filter p xs) + length (filter p ys)
```

holds for all lists `xs` and `ys`.

a) What *precisely* should we prove by induction? State a property P(`xs`), including possible quantifiers, so that proving this property by induction implies the (above) goal of this exercise.

b) For your chosen property P, state (including all possible quantifiers) and prove the base case goal.

c) State (including all possible quantifiers) the inductive hypothesis of the proof.

d) State (including all possible quantifiers) and prove the step case goal.

e) Explain using no more than 150 words how the base case and step case indeed prove the required property for all values of xs. You may reference induction principles, types and your property `P` among other things. (Note that this must be typed and submitted separately to turnitin)

In all proofs indicate the justification (e.g. the line of a definition used) for each step. You may assume basic properties of the `max` function that computes the maximum of two numbers that you can justify with `math`.

Question 2 **Structural Induction on Trees** [3 + 7 + 3 + 4 + 8 Credits]

Consider the following definition of ternary trees of an arbitrary type `a`:

```
data TerTree a = Leaf a | Node a (TerTree a) (TerTree a) (TerTree a)
```

together with the functions `height` and `size` that compute the height, and the number of nodes in a tree, respectively.

```
height :: TerTree a -> Int
height (Leaf x) = 0                                                -- H1
height (Node x l m r) = 1 + maxThree (height l) (height r) (height m)  -- H2

size :: TerTree a -> Int
size (Leaf x) = 1                                       -- S1
size (Node x l m r) = 1 + size l + size m + size r      -- S2
```

In order to do this, a new max function has been defined to calculate the max of three elements (`maxThree`), that is

```
maxThree :: (Ord a) => a -> a -> a -> a
maxThree x y z = max (max x y) z       -- M1
```

The type `TerTree` is an extension of the binary tree that we learnt in the lectures. Here each node has three children instead of two. So a non-leaf element of this type looks like `Node a l m r`, where a is the node value, `l` is the left child, `m` is the middle child, and `r` is the right child. While the leaves would take the form `Leaf x`.

Given that all elements of type `TerTree a` have to be constructed in one of those ways, this question is about identifying the correct induction principle for `TerTree`s and using it to show that the property

$$P(\texttt{t}) \equiv \texttt{size t} \leq \frac{3^{(\texttt{height t})+1} - 1}{2}$$

holds for all elements `t` of type `TerTree a`.

a) Propose an induction principle that allows us to prove that a property $P(\texttt{t})$ holds for all elements `t` of type `TerTree a`?

   State the premises P_1 and P_2, including **all** necessary quantifiers, so that $\frac{P_1 \quad P_2}{\forall \texttt{t}.P(\texttt{t})}$ is a valid induction principle for TerTrees.

b) Briefly justify your answer for (a), use no more than 300 words. Make sure to discuss at least the following points:

   - Why the premises you defined are sufficient for the required principle. I.e., explain why proving P_1 and P_2 is enough to conclude the property holds for all TerTrees.
   - What modifications you have made to the premises of the binary tree induction principle, and why you decided that these were required.

   You must submit this part as a typed file into turnitin.

To show that $P(\texttt{t})$ holds for all elements `t` of type `TerTree a`, we use structural induction (and the induction principle identified above in (a)).

c) State (including possible quantifiers) and prove the base case goal.

d) State (including possible quantifiers) the inductive hypotheses of the proof.

e) State (including possible quantifiers) and prove the step case goal.

In all proofs indicate the justification (e.g. the line of a definition used) for each step. You may assume basic properties of the `maxThree` function that computes the maximum of three numbers that you can justify with `math` (and no further justification is needed for that step).

Question 3 **Hoare Logic (partial correctness)** [5 + 15 + 5 Credits]

The program below, called `foo`, calculates $x * y$ and stores the value in r ('%' is the modulo operation, e.g. 345 % 10 = 5, and '/' is integer division – division without reminder, e.g. 345 / 10 = 34):

```
      while (m > 0)
          if m%2=1 then
                r := r + n;
foo             m := m - 1;          body
          else
                n := n * 2;
                m := m / 2;
```

(You may use the abbreviation ‘`body`’ to refer to the program inside the while loop, and ‘`foo`’ to refer to the entire program.)

The goal of this exercise is to show that `foo` conforms to its specification, i.e. to show that the Hoare triple

$$\{(m = x) \land (n = y) \land (r = 0) \land (x \geq 0)\}\ \texttt{foo}\ \{(r = x * y)\}$$

is valid.

a) Find a suitable loop invariant I. Here *suitable* means

1. The invariant is established: $(m = x) \land (n = y) \land (r = 0) \land (x \geq 0) \to I$.
2. The postcondition is established: $I \land \neg b \to (r = x * y)$.
3. The invariant is invariant: $\{I \land b\}\texttt{body}\{I\}$.

where $b \equiv m > 0$ is the loop condition. State the invariant, no proof required (yet).

b) Using invariant I found in (a), or otherwise, prove that

$$\{I \land b\}\ \texttt{body}\ \{I\}$$

is valid in the Hoare calculus for partial correctness. Justify each step in your proof.

c) Continue the proof in (b) to prove that

$$\{(m = x) \land (n = y) \land (r = 0) \land (x \geq 0)\}\ \texttt{foo}\ \{(r = x * y)\}$$

is valid in the Hoare calculus for partial correctness. Justify each step in your proof.

Question 4 **Hoare Logic (total correctness)** [5 + 15 + 5 Credits]

Consider the (same) program `foo`:

```
          while (m > 0)
              if m%2=1 then
                  r := r + n;
foo               m := m - 1;        body
              else
                  n := n * 2;
                  m := m / 2;
```

We now only want to prove that this program *terminates*, i.e. *without* making guarantees about the variables after execution. In other words, we want to show that the triple

$$[\mathit{True}]\ \texttt{foo}\ [\mathit{True}]$$

is provable in the Hoare calculus for total correctness.

a) Identify and state a suitable *invariant* I and *variant* E for the loop. Here *suitable* means

1. The invariant is established: $\mathit{True} \to I$
2. The postcondition is established: $I \land \neg b \to \mathit{True}$
3. The variant is positive: $I \land b \to E \geq 0$
4. The variant decreases, and the invariant is invariant: $[I \land b \land E = k]\ \texttt{body}\ [I \land E < k]$

where $b \equiv m > 0$ is the loop condition. State the variant and invariant, no proof is required (yet).

b) Using your variant E and invariant I in (a) give a proof of the Hoare triple

$$[I \land b \land E = k]\ \texttt{body}\ [I \land E < k]$$

in the Hoare calculus for total correctness. Justify each step of your proof.

c) Continue the proof in (b) to prove that

$$[\mathit{True}]\ \texttt{foo}\ [\mathit{True}]$$

in the Hoare calculus for total correctness. Justify each step of your proof.

1 Appendix — Hoare Logic Rules (partial correctness)

- Assignment:

$$\{Q(e)\}\ \texttt{x} := \texttt{e}\ \{Q(x)\}$$

- Precondition Strengthening:

$$\frac{P_s \rightarrow P_w \qquad \{P_w\}\ S\ \{Q\}}{\{P_s\}\ S\ \{Q\}}$$

  You can always replace predicates by equivalent predicates,
  i.e. if $P_s \leftrightarrow P_w$; just label your proof step with ‘precondition equivalence’.

- Postcondition Weakening:

$$\frac{\{P\}\ S\ \{Q_s\} \qquad Q_s \rightarrow Q_w}{\{P\}\ S\ \{Q_w\}}$$

  You can always replace predicates by equivalent predicates,
  i.e. if $Q_s \leftrightarrow Q_w$; just label your proof step with ‘postcondition equivalence’.

- Sequence:

$$\frac{\{P\}\ S_1\ \{Q\} \qquad \{Q\}\ S_2\ \{R\}}{\{P\}\ S_1; S_2\ \{R\}}$$

- Conditional:

$$\frac{\{P \wedge b\}\ S_1\ \{Q\} \qquad \{P \wedge \neg b\}\ S_2\ \{Q\}}{\{P\}\ \textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \{Q\}}$$

- While Loop:

$$\frac{\{I \wedge b\}\ S\ \{I\}}{\{I\}\ \textbf{while}\ b\ \textbf{do}\ S\ \{I \wedge \neg b\}}$$

2 Appendix — Hoare Logic Rules (total correctness)

- Assignment:

$$[Q(e)]\ \texttt{x := e}\ [Q(x)]$$

- Precondition Strengthening:

$$\frac{P_s \ \rightarrow\ P_w \qquad [P_w]\ S\ [Q]}{[P_s]\ S\ [Q]}$$

You can always replace predicates by equivalent predicates,
i.e. if $P_s \leftrightarrow P_w$; just label your proof step with 'precondition equivalence'.

- Postcondition Weakening:

$$\frac{[P]\ S\ [Q_s] \qquad Q_s \ \rightarrow\ Q_w}{[P]\ S\ [Q_w]}$$

You can always replace predicates by equivalent predicates,
i.e. if $Q_s \leftrightarrow Q_w$; just label your proof step with 'postcondition equivalence'.

- Sequence:

$$\frac{[P]\ S_1\ [Q] \qquad [Q]\ S_2\ [R]}{[P]\ S_1; S_2\ [R]}$$

- Conditional:

$$\frac{[P \wedge b]\ S_1\ [Q] \qquad [P \wedge \neg b]\ S_2\ [Q]}{[P]\ \textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ [Q]}$$

- While Loop (Total Correctness):

$$\frac{I \wedge b \rightarrow E \geq 0 \qquad [I \wedge b \wedge E = n]\ S\ [I \wedge E < n]}{[I]\ \textbf{while}\ b\ \textbf{do}\ S\ [I \wedge \neg b]}$$

where n is an auxiliary variable not appearing anywhere else.