NOTE TO STUDENTS: This file contains sample solutions to the term test together with the marking scheme and comments for each question. Please read the solutions and the marking schemes and comments carefully. Make sure that you understand why the solutions given here are correct, that you understand the mistakes that you made (if any), and that you understand *why* your mistakes were mistakes.

Remember that although you may not agree completely with the marking scheme given here it was followed the same way for all students. We will remark your test only if you clearly demonstrate that the marking scheme was not followed correctly.

For all remarking requests, please submit your request **in writing** directly to your instructor. For all other questions, please don't hesitate to ask your instructor during office hours or by e-mail.

Question 1. [6 MARKS]

Consider the sequence of "Fibonacci Numbers" defined on the right:

$$F_0 = 0,$$
$$F_1 = 1,$$
$$F_n = F_{n-1} + F_{n-2} \quad \forall n \geqslant 2.$$

Use simple induction to prove that for all integers $n \geqslant 0$, $F_0^2 + F_1^2 + \cdots + F_n^2 = F_n F_{n+1}$.

SAMPLE SOLUTION:

Base Case: $F_0^2 = 0 = 0 \cdot 1 = F_0 F_1$.

Ind. Hyp.: Assume $n \geqslant 0$ and $F_0^2 + F_1^2 + \cdots + F_n^2 = F_n F_{n+1}$.

Ind. Step:

$$\begin{aligned} F_0^2 + F_1^2 + \cdots + F_n^2 + F_{n+1}^2 &= F_n F_{n+1} + F_{n+1} F_{n+1} \qquad \text{(by I.H.)} \\ &= (F_n + F_{n+1}) F_{n+1} \\ &= F_{n+2} F_{n+1} \qquad \text{(by definition of } F_{n+2} \text{ for } n \geqslant 0) \\ &= F_{n+1} F_{n+2} \end{aligned}$$

MARKING SCHEME:

- **Structure** [3 marks]: answer clearly contains a base case, a simple induction hypothesis, and an induction step that makes use of the induction hypothesis — even if none of these are labelled explicitly
- **Content** [3 marks]: correct proof of the base case [0.5], correct induction hypothesis [0.5], and correct proof of the induction step [2]

MARKER'S COMMENTS:

- **error code S1.1** [−0.5]: Your predicate is too strong and "overwrites" its argument (*e.g.*, $P(n) = \forall n, \ldots$).
- **error code S1.2** [−0.5]: Your predicate is used in a non-boolean context (*e.g.*, $P(n) = F_n F_{n+1}$).
- **error code C1.1** [−0.5]: Your induction hypothesis does not link up with your base case(s), *e.g.*, your prove bases cases for $n = 0$ and $n = 1$ but then assume $n \geqslant 2$ in your induction hypothesis.
- **error code C1.2** [−0.5]: Your induction hypothesis is too strong (*e.g.*, "assume $P(n)$ for all $n \ldots$").
- **error code C2.1** [up to −2]: The algebra in your induction step is incorrect.
- **error code C2.2** [up to −2]: Your induction step is missing too many steps.
- **error code C2.3** [up to −2]: The algebra in your induction step is confusing and unclear.
- **common error**: Many students thought they needed n to be at least 2 in their induction step, but that is not the case.

Question 2. [8 MARKS]

For $x, y \in \{\mathsf{True}, \mathsf{False}\}$, let "$x \oplus y$" denote the *exclusive-or* of x and y, which is defined to be True iff exactly one of x and y is True. (The full truth table for $\oplus$ is shown on the right.)

True $\oplus$ True	= False
True $\oplus$ False	= True
False $\oplus$ True	= True
False $\oplus$ False	= False

Use complete induction to prove that for all integers $n \geqslant 0$, every propositional formula F that contains n occurrences of the $\oplus$ connective (and no other connective) is True iff F contains an *odd* number of True variables.

(For example, $(x_2 \oplus x_5) \oplus ((x_1 \oplus x_4) \oplus x_3)$ is False when x_1, x_3 are True and x_2, x_4, x_5 are False, but the same formula is True when x_1, x_3, x_4 are True and x_2, x_5 are False. Note that a propositional formula with *no* connective is simply equal to some propositional variable.)

SAMPLE SOLUTION:

Base Case: If F is a propositional formula that contains 0 occurrences of $\oplus$, then F is a single variable, so F is True iff it contains one True variable.

Ind. Hyp.: Assume $n > 0$ and every propositional formula that contains *fewer* than n occurrences of $\oplus$ is True iff it contains an odd number of True variables.

Ind. Step: Suppose F is a propositional formula that contains n occurrences of $\oplus$.

Since $n > 0$, $F = F_1 \oplus F_2$ where F_1 contains $n_1 \geqslant 0$ occurrences of $\oplus$ and F_2 contains $n_2 \geqslant 0$ occurrences of $\oplus$ and $n = n_1 + n_2 + 1$ (so $n_1 < n$ and $n_2 < n$).

By the I.H., F_1 is True iff it contains an odd number of True variables and F_2 is True iff it contains an odd number of True variables.

If the numbers of True variables in F_1 and F_2 have the same parity (both even or both odd), then F_1 and F_2 have the same value so F is False, and the number of True variables in F is even.

If the numbers of True variables in F_1 and F_2 have different parity (one even and one odd), then F_1 and F_2 have different values so F is True, and the number of True variables in F is odd.

In either case, F is True iff it contains an odd number of True variables.

MARKING SCHEME:

- **Structure** [4 marks]: answer clearly contains base case(s) [1], a complete induction hypothesis [1], and an induction step that makes use of the induction hypothesis [2] — even if none of these are labelled explicitly
- **Content** [4 marks]: correct proof of the base case(s) [0.5], correct induction hypothesis [0.5], and correct proof of the induction step [3]

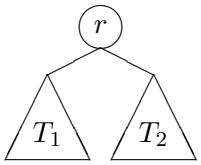
MARKER'S COMMENTS:

- **common error** [−0.5]: proving base case for $n = 1$ instead of $n = 0$
- **common error** [−2]: splitting on the last connective, *i.e.*, the inductive step only applies to formulas of the form $F \oplus x$ for a formula F and variable x
- **common error** [−1.5]: no base case at all
- **common error** [−1.5]: not properly combining F_1 and F_2 in the inductive step

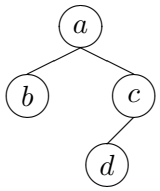
Question 3. [12 MARKS]

Consider the following recursive definition of *binary trees*.

- The *empty tree* (that contains **no** node and **no** edge) is a binary tree.
- If T_1 and T_2 are binary trees and r is a single node, then the tree obtained by making T_1 the left subtree of r and T_2 the right subtree of r is also a binary tree. (This is illustrated on the right.)
- Nothing else is a binary tree.



A *path* is a sequence of nodes with edges between them. For example, "b—a—c—d" is a path between b and d in the binary tree pictured on the right. A sequence with just one node (like "c") is a valid path, and so is the empty sequence "" (vacuously).



Use structural induction to prove that for all binary trees T, there is a unique path between any two nodes of T.

SAMPLE SOLUTION:

Base Case: Vacuously, there is a unique path between any two nodes in the empty binary tree.

Ind. Hyp.: Suppose that T_1 and T_2 are binary trees such that there is a unique path between any two nodes in T_1 and there is a unique path between any two nodes in T_2. Suppose also that r is a node.

Ind. Step: Consider the binary tree T formed by root r with left subtree T_1 and right subtree T_2.

Let x and y be any two nodes in T.

If $x = y = r$, then "r" is the unique path "between" x and y.

If x and y belong to the same subtree (T_1 or T_2), then there is a unique path between x and y, by the I.H.

If $x = r$ and y belongs to T_1 (or any other situation where one of x, y is equal to r and the other belongs to T_1 or T_2), then there is a unique path between y and the root of T_1, by the I.H. Together with the edge between the root of T_1 and r, this forms a unique path between x and y.

If x belongs to T_1 and y belongs to T_2 (or vice-versa), then by the I.H., there is a unique path between x and the root of T_1, and a unique path between y and the root of T_2. These paths are joined by the edges from r to the root of each subtree, to form a unique path between x and y.

In every case, there is a unique path between x and y.

MARKING SCHEME:

- **Structure** [6 marks]: answer clearly contains a base case, an induction hypothesis, and an induction step that makes use of the induction hypothesis, all structured following the recursive definition of binary trees — even if none of these are labelled explicitly
- **Content** [6 marks]: correct proof of the base case, correct induction hypothesis, and correct proof of the induction step (including correct handling of the various cases)

MARKER'S COMMENTS:

- **error code NOIH**: The proof is missing an inductive hypothesis.
- **error code BIH0**: Bad inductive hypothesis: you assumed $\forall T, P(T)$, which is what you set out to prove.
- **error code BIH1**: Bad inductive hypothesis: it is missing components. You forgot to assume that your predicate holds for the two subtrees from which you construct the larger one. The inductive hypothesis also needs to say that the variables used to represent the subtrees are in the set of binary trees.
- **error code CASES**: You missed some of the cases. These cases include:
  - Both nodes being in the same subtree.
  - Both nodes being the root.
  - One node being the root and the other in one of the subtrees.
  - Each node being in a different subtree.
- **error code JUST**: You did not properly justify how a path from a node in the subtree to the root of the new tree is unique. Specifically, you need to reason that the root of a subtree is connected to the root of the whole tree by a unique edge (assuming the subtree is not empty). You then need to reason that by the inductive hypothesis there exists a unique path from a node in a subtree to the root of the subtree. Using the unique edge to the root, we then get a unique path to the root.
- **error code BASE**: Incorrect base case. The base case in structural induction must prove the predicate for the basic elements, which in this case is the empty tree.
- **error code WST**: Wrong structure for the proof. You needed to use structural induction rather than complete/simple induction.
- **common error** [no penalty]: Some people had a predicate that talked about a binary tree T, but instead they wrote $P(n)$.
- **common error** [no penalty]: Some people did the case of having the two nodes in separate subtrees but not where the beginning/end node is the root. Most of these people showed how to get from a subtree to the root, and in these cases I did not penalize them for not explicitly saying it was a separate case.

Bonus. [3 MARKS]

Use the Principle of Well Ordering to prove that for all positive integers n,

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}.$$

(*Attempts to prove this by induction will receive* ***no*** *credit.*)

SAMPLE SOLUTION: (Not provided for bonus...)

MARKING SCHEME: *Be particularly picky when marking the bonus!*

- **Structure** [2 marks]: clear and correct structure for a proof using well ordering: assume the statement is false, consider the set of counter-examples (not empty by assumption), use well ordering to get the smallest counter-example k, show a contradiction for every possible value of k
- **Content** [1 mark]: correct cases considered for the values of k, correct contradiction reached in every case, correct algebra and generally well-written proof