Quiz 1 Solutions

Problem 1. Use induction to prove our proposition, P(n),

$$1^3 + 2^3 + \ldots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$$

whenever n is a natural number.

Base Case: $n = 0$

The left side is an empty sum, so $0 = \left(\frac{0 \cdot 1}{2}\right)^2 = 0$.

Inductive Step: Show that $P(n) \rightarrow P(n+1)$.

By the inductive hypothesis, we assume that

$$1^3 + 2^3 + \ldots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$$

$$\begin{aligned}
&1^3 + 2^3 + \ldots + n^3 + (n+1)^3 \\
&\qquad = \left(\frac{n(n+1)}{2}\right)^2 + (n+1)^3 \\
&\qquad = \frac{n^2(n+1)^2 + 4(n+1)(n+1)^2}{4} \\
&\qquad = \frac{(n^2 + 4n + 4)(n+1)^2}{4} \\
&\qquad = \left(\frac{(n+1)(n+2)}{2}\right)^2
\end{aligned}$$

And our hypothesis is proved.

Common errors

The most common error was to assume that the hypothesis was true, and then to prove it by reducing both sides of the equation to something that is obviously equal. This is not a good proof technique.

$$\begin{aligned}
1^3 + 2^3 + \ldots + n^3 + (n+1)^3 &= \left(\frac{(n+1)(n+2)}{2}\right)^2 \\
\left(\frac{n(n+1)}{2}\right)^2 + (n+1)^3 &= \left(\frac{(n+1)(n+2)}{2}\right)^2 \\
\frac{n^2(n^2 + 2n + 1)}{4} + n^3 + 3n^2 + 3n + 1 &= \frac{(n^2 + 3n + 2)^2}{4} \\
\frac{(n^4 + 2n^3 + n^2) + (4n^3 + 12n^2 + 12n + 4)}{4} &= \frac{n^4 + 3n^3 + 2n^2 + 3n^3 + 9n^2 + 6n + 2n^2 + 6n + 4}{4} \\
\frac{n^4 + 6n^3 + 13n^2 + 12n + 4}{4} &= \frac{n^4 + 6n^3 + 13n^2 + 12n + 4}{4}
\end{aligned}$$

A technique which is similar but which is not at all correct is to start with the equation you wanted to prove (ie, $f(n) = g(n)$) and then perform the same operations on both sides of the equation until they're equal.

$$\begin{aligned} n^2 + 3n + 1 &= n^2 + 3n + 1 \\ 3n + 1 &= 3n + 1 \\ 1 &= 1 \end{aligned}$$

This is OK for a quick check, but does not quite work as a proof, since it is possible to get equality at the end even if the starting sides of the equation were not equal. For example

$$\begin{aligned} n^2 + 3n + 1 &= 2n^2 + 5n + 8 \\ 0 \cdot (n^2 + 3n + 1) &= 0 \cdot (2n^2 + 5n + 8) \\ 0 &= 0 \end{aligned}$$

Remember that in an ideal direct proof, each step should follow from the previous one, until the final step which is what we want to prove.

Problem 2. Number theory.

(a) Find an integer x such that

$$384 \cdot x = 4 \pmod{566}.$$

Show your derivation.

The following is what we get from Euclid's algorithm:

$182 = 566 - 384$
$20 = 384 - 2 \cdot 182$
$2 = 182 - 9 \cdot 20$
$0 = 20 - 10 \cdot 2$

Omitting the last line and doing some substitutions to put each RHS in terms of 566 and 384:

$182 = 566 - 384$
$20 = 384 - 2 \cdot 182 = 384 - 2(566 - 384) = -2 \cdot 566 + 3 \cdot 384$
$2 = 182 - 9 \cdot 20 = (566 - 384) - 9(-2 \cdot 566 + 3 \cdot 384) = 19 \cdot 566 - 28 \cdot 384$

So $4 = 2 \cdot 2 = 38 \cdot 566 - 56 \cdot 384$. This means $-56 \cdot 384 \equiv 4 \pmod{566}$. We can reverse the order of multiplication, of course, so $x = -56$ is a solution. Note that an integer is a solution iff its product with 384 differs from $-56 \cdot 384$ by a multiple of 566, which happens iff it differs from -56 by a multiple of 283. (A factor of two is picked up in multiplication by 384). Thus another solution is $x = 227$.

(b) Why can't you find an integer x for which

$$384 \cdot x = 1 \pmod{566}?$$

This is equivalent to finding an integer x such that there is some integer y for which $384 \cdot x + 566 \cdot y = 1$. Since 384 and 566 are even, the LHS must be even, so no integers can give equality to 1.

Another way to say this is to note that we proved in class that every number of the form $384 \cdot x + 566 \cdot y$ is a multiple of the GCD of 384 and 566, which is 2.

Problem 3. Prove the set identity

$((A - B) \cup (B - A))^c = (A^c \cup B) \cap (A \cup B^c)$

Using the definition of set difference, both of De Morgan's Laws, and the commutativity of union (in that order):

$$\begin{aligned}((A - B) \cup (B - A))^c &= ((A \cap B^c) \cup (B \cap A^c))^c \\ &= (A \cap B^c)^c \cap (B \cap A^c)^c \\ &= (A^c \cup B) \cap (B^c \cup A) \\ &= (A^c \cup B) \cap (A \cup B^c)\end{aligned}$$

Alternatively, one can use the definitions of the operations. We need to prove x is in the LHS if and only if it is in the RHS.

$x \in ((A - B) \cup (B - A))^c$ means that $x \notin (A - B) \cup (B - A)$. This means that $x \notin A - B$ and $x \notin B - A$.

So consider cases. If $x \in A$, then certainly $x \in A \cup B^c$. Furthermore, since by assumption $x \notin A - B$, we must have $x \in B$. Thus, $x \in B \cup A^c$. Putting these together shows $x \in (A \cup B^c) \cap (B \cup^c)$. Now consider the other case. If $x \notin A$, meaning $x \in A^c$, then certainly $x \in A^c \cup B$. Also, (since $x \notin B - A$), we must have $x \notin B$. That is, $x \in B^c$. Therefore, $x \in B^c \cup A$. Thus, $x \in (A^c \cup B) \cap (B^c \cup A)$.

This proves one direction; the other is similar.

Problem 4. A *full binary tree* is either

- A *leaf* node with no subtrees
- A *non-leaf* node with a left and a right subtree, each of which is a full binary tree.

(a) Draw all full binary trees with 5 or fewer nodes.

(b) Prove that for any full binary tree, the number of non-leaf nodes is exactly one less than the number of leaves.

Solution 1: Let,

$$\begin{aligned} P(n) \quad = \quad & \text{“For a full binary tree with n non-leaf nodes} \\ & n = l - 1 \text{ where } l \text{ is the number of leaves”} \end{aligned}$$

Proof by strong induction on the number of non-leaves, n:

Base Case: $n = 0$, tree is made up of exactly one node. $P(0)$ is true since a node with no subtrees is a leaf ($l = 1$) and thus $0 = 1 - 1$.

Inductive Hypothesis: For $n \leq k$ non-leaf nodes assume $P(n)$ is true, that is, “For a full binary tree with $n \leq k$, the non-leaf nodes $n = l - 1$”.

Inductive Step: Prove P(n=k+1): For a full binary tree with $k+1$ non-leaf nodes, the number of non-leaf nodes in the tree by definition is made up of:

- one root node,
- the non-leaf nodes in the left-subtree of the root, n_{left}, and
- the non-leaf nodes in the right-subtree of the root, n_{right}.

The number of leaves, l_{tree} in the tree by definition is made up of:

- the number of leaves in the left-subtree of the root, l_{left}, and
- the number of leaves in the right-subtree of the root, l_{right}.

$$\begin{aligned}
k+1 &= 1 + n_{left} + n_{right} && \text{by the definition of full binary tree} \\
&= 1 + (l_{left} - 1) + (l_{right} - 1) && \text{by the inductive hypothesis} \\
&= 1 + (l_{left} + l_{right}) - 1 - 1 && \\
&= l_{tree} - 1 && \text{by definition of full binary tree}
\end{aligned}$$

Solution 2:

We use induction on the given recursive definition.

There is only one base case, which is a leaf node with no subtrees. So there is one leaf node and no non-leaf nodes, satisfying the statement.

There is only one combination case, which is a non-leaf node (call it v) with a left and a right subtree, each of which is a full binary tree. By the IH, each of these subtrees has exactly one fewer non-leaf nodes than leaf nodes. Taken together, there are then two fewer non-leaf nodes than leaf nodes. Since this accounts for all of the nodes in the tree except for v (which is a non-leaf node), the tree has exactly one fewer non-leaf nodes than leaf nodes.

Errors: The most common error in this problem was to use facts about full binary trees that were not at all obvious from the above definition. For example, some people said during the inductive step

- "find a node with two children that are leaves; take away the leaves; this gives a smaller full binary tree to which an inductive hypothesis applies."
- "find a leaf turn it into a non-leaf by adding two children; the new tree has $k+1$ non-leaf nodes, while the original tree has k non-leaf nodes to which the inductive hypothesis can be applied"

Unfortunately, it is not at all obvious from the above definition of a full binary tree that that taking away or adding two leaf nodes results in a full binary tree. While this may be true, it is something that would have to be proven.

Main point is, that when you do induction on a recursive definition, your inductive step needs to be based on the combination rules of the inductive definition, and not on other things that are "obvious" from drawing pictures.

Problem 5. Use the following tables to describe the properties of the various relations and functions. Write yes or no in each box. Don't give proofs (it'd be too hard to squeeze them into the boxes). **Pay careful attention** to the specified domains and ranges.

(a) The following relations:

- $R = \{(a, b) \in (\mathbb{N} - \{0\})^2 \mid a \text{ divides } b\}$
- $S = \{(X, Y) \in (2^{\mathbb{N}} - \{\emptyset\})^2 \mid X \cap Y \neq \emptyset\}$
- $T = \{(x, y) \in \mathbb{R} \times \mathbb{R} \mid x + y = 0\}$

(Recall that antisymmetric means that $a \sim b$ and $b \sim a$ imply $a = b$).

relation	reflexive	symmetric	antisymmetric	transitive	function
$R:\ a \mid b$	yes	no	yes	yes	no
$S:\ X \cap Y \neq \emptyset$	yes	yes	no	no	no
$T:\ x + y = 0$	no	yes	no	no	yes

(b) The following functions:

- $f : \mathbb{N} \to \mathbb{N},\ f(x) = 5x.$
- $g : \mathbb{R}^+ \to \mathbb{N},\ g(x) = \lfloor x \rfloor.$
- $h : \mathbb{R} \to \mathbb{R}^+,\ h(x) = e^x.$

Give an inverse if one exists (under the given domain and range).

function	injective	surjective	inverse
$f(x) = 5x$	yes	no	none
$g(x) = \lfloor x \rfloor$	no	yes	none
$h(x) = e^x$	yes	yes	$\ln(x)$

Errors: A common error here was not to pay attention to the domain and range. Although $f(x) = 5x$ is a bijection from the reals to the reals, it is not a bijection from the naturals to the naturals because there is no *natural* number that satisfies $5x = 1$.

Problem 6. Consider the following algorithm, which takes natural numbers a and b, with $b > 0$, and returns an ordered pair of natural numbers.

Algorithm Div(a, b)

```
if (a < b)
    return (0, a)
else
    (q, r) ←Div(a − b, b)
    return (q + 1, r)
```

(a) Prove Div terminates in at most $a + 1$ steps.

We use strong induction on a. The inductive hypothesis is "if Div(a', b) terminates for all $a' < a$, then Div(a, b) terminates".

Base Case: $a < b$. Then Div returns immediately (that is, in one step), which is certainly no more than $a + 1$ steps, since $a \geq 0$.

Inductive step: $a \geq b$. Then $0 \leq a - b < a$, since $b > 0$. In this case, there is a recursive call Div$(a - b, b)$. By the IH, this recursive call returns in at most $a - b + 1$ steps. Since $a - b < a$, this is at most $a + 1$ steps.

Common errors

Please note that many people received credit for a base case of $a = 0$, which is actually wrong. This case is only reached when a is a multiple of b, so the induction doesn't stand, since for cases where a does not divide b, no base case has been shown.

We insisted that you use induction for this because any sort of argument that goes something like "And then it keeps on subtracting X until..." is not rigorous. Such an argument is begging for induction.

(b) Prove that Div returns the quotient and remainder of a on division by b.

Recall that the quotient and remainder of a on division by b are defined to be the unique integers q and r such that $a = qb + r$ and $0 \leq r < b$. These integers must exist by the Division Algorithm.

We use strong induction on a.

Base case: $a < b$. Then Div immediately returns $(0, a)$. Since $a = 0 \cdot b + a$ and $0 \leq a < b$, these are the quotient and remainder of a on division by b.

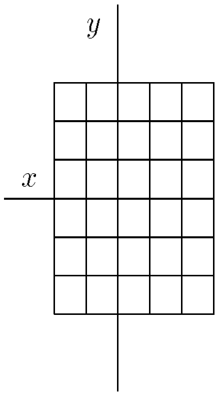
Inductive step: $a \geq b$. Then $0 \leq a - b < a$. In this case, there is a recursive call Div$(a - b, b)$. By the IH, this recursive call returns the quotient and remainder of $a - b$ on division by b. So $a - b = qb + r$ and $0 \leq r < b$. Then $a = (q + 1)b + r$ and $0 \leq r < b$. So $q + 1$ and r are the quotient and remainder of a on division by b, and these are returned by Div.

Common errors

Some people tried to come up with an invariant for this problem. An invariant is used for proving program correctness when there is some state which exists *outside* of the procedures which make up that program. We argue that if this state satisfies some criteria (the invariant) upon entering the procedure, it must also satisfy it on exiting the procedure for the code to be correct. Correctness using invariants has an implicit induction within it which says that the base case is that the invariant holds after n=0 procedure calls (ie, the initial conditions satisfy the invariant), and if the invariant holds after n procedure calls and if all the procedures keep the invariant, then it will hold after n+1 procedure calls.

Note that this is very different from what's happening here. We're not verifying a program, but a procedure. And the state here is not outside of the procedure (for example, not a global variable), but we're interested in the values of q and r within the program and especially what Div(a,b) *returns*. An invariant involving q and r makes no sense, since it couldn't hold upon entry since q and r are local variables and their values upon entry are undefined.

Problem 7. David is located at position $(1, 0)$ on the two-dimensional $(\mathbb{Z} \times \mathbb{Z})$ x-y plane. His house is located at the origin. David wants to go home. But it is windy, and each time David takes a one-unit step east or west, he also gets blown a one-unit step north or south (he can pick which).



(a) On the grid above, mark with X's all the points that David can reach in one step. Mark with O's the points that David can reach in two steps. (The axes labeled x and y intersect at David's house.)

(b) Give an informal explanation for why David can never get home.

David starts off with the sum of his coordinates $(1 + 0 = 1)$ odd. Every time he moves, the parity of each coordinate changes, so the sum of his coordinates remains odd. Since the sum of the coordinates of his home $(0 + 0 = 0)$ is even, he can never reach it.

(c) Describe a state machine that represents David's motion, and list its transitions.

We introduce state variables $x, y \in \mathbb{Z}$. We designate exactly one start state: the one for which $x = 1$ and $y = 0$. The transitions are as follows:

move($go, blow$)
  Precondition: $go, blow \in \{-1, 1\}$
  Effect: $x \leftarrow x + go$
    $y \leftarrow y + blow$

Getting the notation for transitions wrong was a common error. In particular, some people wrote program code that returned values. The fragments we use for transitions reflect changes to state variables, which are best represented by assignment operations. The role of the Effect clause is to *describe* what the value of the state variables will be after the transition. We have adopted an assignment style of notation for the Effect clause to accentuate the fact that state variables are changed by the transition.

(d) Prove that David can never get home.

We use the following invariant: $x + y$ is odd for all reachable states. This is true for the start state $(0 + 1 = 1)$. For each transition, go and $blow$ are each odd, so their sum is even. Thus each transition increases $x + y$ by an even amount $(go + blow)$, preserving the invariant.

Since $x + y$ is even for the state corresponding to home, it is not a reachable state of our state machine. Thus David can never get home.

Errors: There were two main types of errors in coming up with invariants.

One was to provide a correct statement that was not amenable to an inductive proof. For example, the fact that $x \neq y$ is true of all reachable states, but is not preserved by all transitions (you can get from (2,0) to (1,1), for instance). This means that it is not really possible to prove it by induction because there are certain states (such as $(2, 0)$) where the inductive step doesn't work. Note that writing an inductive proof does not strictly require that the property be preserved for all transitions (only those starting from reachable states). As a practical matter, however, it is hard to make the inductive step take advantage of the fact that the first step was reachable. (This is because reachability is not a property of the state in isolation but depends on the *entire* state machine definition). Thus one has little choice but to show in the inductive step that *any* state (reachable or not) that satisfies the invariant has outgoing transitions leading only to states that satisfy the invariant.

Another mistake was to prove an invariant that is true but does not imply what we want. Some proved that the total distance travelled was always a multiple of $\sqrt{2}$. While this invariant is indeed true, it does not prove what we want because *straight-line* distance from the starting point does not always have to be a multiple of $\sqrt{2}$, so the above fact does not amount to a proof.

Problem 8. Let F be a function from A to B and let $G = F^{-1}$ be its inverse relation (G is a relation from B to A, but not necessarily a function). Prove that $G \circ F$ is an equivalence relation on A.

By definition, x is related to y by $G \circ F$ iff there is a $z \in B$ such that x is related to z by F and z is related to y by G. That is, $z = F(x)$ and $z = F(y)$. So x is related to y by $G \circ F$ iff $F(x) = F(y)$.

Reflexitivity: Let $x \in A$. Then $F(x) = F(x)$, so $(x, x) \in G \circ F$.

Symmetry: Let $x, y \in A$ such that $(x, y) \in G \circ F$. Then $F(x) = F(y)$ (and $F(y) = F(x)$). So $(y, x) \in G \circ F$.

Transitivity: Let $x, y, z \in A$ such that $(x, y), (y, z) \in G \circ F$. Then $F(x) = F(y)$ and $F(y) = F(z)$. So $F(x) = F(z)$ and $(x, z) \in G \circ F$.

One common mistake for this problem was to assume that $G \circ F$ is the identity function. In general, G is not a function, and neither is $G \circ F$.

Problem 9. Consider the following algorithm that takes an array (list) $A[1 \ldots n]$ of numbers such that $A[1] < A[2] < \cdots < A[n]$, and a number q, and is supposed to find q if it is present in the list.

Find(A, n, q)

```
i ← 1
j ← n + 1
While (j − i > 1)
    choose some k with i < k < j
    if A[k] ≤ q
        i ← k
    else
        j ← k
Return i
```

(a) prove this function terminates

We first prove (by induction on l) that after the lth pass through the while loop, $j - i \leq n - l$.

Base Case: $l = 0$. When we first reach the while loop, $j - i = n$ because we just set i to 1 and j to $n + 1$.

Inductive Step: By the IH, $j - i \leq n - l$ after the lth pass through the while loop. During the $(l + 1)$st pass, either i or j is moved to a value (that of k) strictly between their values from the end of the lth pass. Thus the $(l+1)$st pass decreases $j - i$ by at least 1. So after the $(l + 1)$st pass through the while loop, $j - i < n - l - 1 = n - (l + 1)$.

The most common error on this part was hand-waving. Informal descriptions of the long-term behavior of the algorithm are not considered proofs by 6.042 standards. Since the reader can safely be assumed to accept induction as a valid proof technique, the above proof *at very least* gives the reader much less room to object to the argument.

The second most common error was to try to prove the desired statement directly by induction. This just doesn't work. If it did, we would not have introduced the more complicated proof technique that we said should be used in this case.

Now suppose that $q = A[r]$ for some r.

(b) State an invariant of the while loop relating i, j, and r. The invariant should be "obviously provable by induction" but you need not prove it.

$i \leq r < j$.

(c) Assuming the invariant, prove that Find returns r.

When the loop terminates, $j - i \leq 1$ ($j \leq i + 1$). Also, the invariant still holds. So $i \leq r < j \leq i + 1$. Since i and r are integers, $i = r$. Since i is then returned, Find returns r.

Surprisingly enough, a number of people didn't even try to make use of the invariant. Use of induction was also common. The invariant deals with exactly the issues that would otherwise require induction. This is why the correctness of the invariant would be proven by induction. Some people apparently didn't read the question and tried to prove the invariant correct instead of proving that Find returns r.

Problem 10. Let p be a prime. Define a relation $R \subseteq \{1, \ldots, p-1\} \times \{1, \ldots, p-1\}$ by aRb if $ab = 1 \pmod{p}$.

(a) Prove that R is a symmetric relation.

Suppose aRb. Then $ab \equiv 1 \pmod{p}$. So $ba \equiv 1 \pmod{p}$ and bRa.

(b) Prove that if $ab = ac \pmod{p}$ for $0 < a < p$, then $b = c \pmod{p}$.

Since $\gcd(a, p) = 1$, there are integers i and k such that $ip + ka = 1$. Thus $ka \equiv 1 \pmod{p}$. Since $ab \equiv ac \pmod{p}$, $kab \equiv kac \pmod{p}$. So $kab \bmod p = kac \bmod p$. But $kab \bmod p = ((ka \bmod p) \cdot b) \bmod p = (1 \cdot b) \bmod p = b \bmod p$ and $kac \bmod p = ((ka \bmod p) \cdot c) \bmod p = (1 \cdot c) \bmod p = c \bmod p$. So $b \bmod p = c \bmod p$. Thus $b \equiv c \pmod{p}$.

(c) Use the previous statement to prove that R is a function which is a bijection.

Lemma 0.1 *For any a in $\{1, \ldots, p-1\}$, there is a unique b in $\{1, \ldots, p-1\}$ such that aRb.*

Proof. Let a in $\{1, \ldots, p-1\}$. Since $\gcd(a, p) = 1$, there are integers i and k such that $ip + ka = 1$. Thus $ka \bmod p = 1 \bmod p$. So $((k \bmod p) \cdot a) \bmod p = 1 \bmod p$. Let b be $(k \bmod p)$. Then $b \in \{1, \ldots, p-1\}$ and $ba \equiv 1 \pmod{p}$. So aRb. By part b, b is the only element of $\{1, \ldots, p-1\}$ that satisfies $aj \equiv 1 \pmod{p}$, so it is the only one that is related to a. ∎

By the lemma, R is a function on $\{1, \ldots, p-1\}$. Using the lemma and part a, we get that every element i of $\{1, \ldots, p-1\}$ is related to by some element of $\{1, \ldots, p-1\}$. So R is surjective. Since the lemma also gives that the element relating to i is unique, R is injective. Thus R is a bijection.

Many students neglected to show that R was a function. First of all, we have specifically asked for a proof that R is a function. Second, we haven't talked about anything other than functions being injective or surjective. So these properties shouldn't be claimed until it is proven that R is a function.

(d) What is R^{-1}?

For arbitrary $a, b \in \{1, \ldots, p-1\}$, part a gives us both that $aRb \Rightarrow bRa$ and $bRa \Rightarrow aRb$. So $aRb \Leftrightarrow bRa$. By definition, $aR^{-1}b \Leftrightarrow bRa$. So $R^{-1} = R$.

(e) Show that if p is not prime, R might not be a function.

As discussed in 2b, there does not exist a solution b to $ab \equiv 1 \pmod{p}$ if $a = 384$ and $p = 566$. For this case, nothing is related to a, so R cannot be a function on the domain $\{1, \ldots, p-1\}$.