There are two components of this problem set: a preliminary question designed to check your understanding of the basic topics covered this week, and a set of more challenging questions designed to make you think critically about the material and apply it in new contexts. Get in the habit of starting work early – the less time you give yourself, the most stressed you’ll find yourself each week!

**Caution:** you must submit two separate files in two separate locations on MarkUs, one for the Preliminary and one for the Challenge.

To avoid suspicions of plagiarism, clearly state any resources (people, print, electronic) outside of your group, the course notes, and the course staff, you consulted at the beginning of your assignment submission.

---

**Preliminary: due January 21, 2014 8:00 pm**

This question is an opportunity for you to check your understanding of the topics and practice writing formal solutions. This is a valuable learning opportunity – if you see that you’re at a loss, get help quickly!

Your goal should not be to get the right answer, but to convince the marker that you know what you’re doing. This question is marked on the following 3-point scale:

<table>
<thead>
<tr>
<th>3: You’ve mastered this topic</th>
<th>1: You don’t really know what you’re doing</th>
</tr>
</thead>
<tbody>
<tr>
<td>2: You’re almost there, but missing something</td>
<td>0: You didn’t submit/had absolutely no clue</td>
</tr>
</tbody>
</table>

This question must be completed INDIVIDUALLY.

Suppose you have an unlimited supply of 3- and 8-cent coins. Recall that for some \( m \in \mathbb{N} \), it is possible to have combinations of these coins make every value greater than or equal to \( m \). You proved this on Problem Set 1 using simple induction.

Reprove this claim here using complete induction. Do not just repeat your proof using simple induction; the goal of this question it to make sure you understand the difference between simple and complete induction. Your proof should be modelled on the following informal reasoning: “First, we know that we can make these values using the coins, and for any higher values we can just add 3-cent coins over and over again from these starting values.”

---

**Challenges: due January 25, 2014 noon**

Answer each question completely, always justifying your claims and reasoning. Your solution will not just be graded on correctness, but on its clarity as well. Technically correct answers that are hard to understand will not receive full marks.

Mark values for each question are contained in the [square brackets].

You may work in groups of up to THREE to complete these questions.

Note: The LaTeX Bonus is in effect for this problem set. See the course webpage for details.

1. [6] For this question, you’ll need to use the definition of the Fibonacci sequence from page 15 of the Course Notes.

   Consider the following sequence of natural numbers \( a_1 = 1, a_2 = 1 \), and for all \( n \geq 3 \), \( a_n = a_{n-1} + a_{n-2} + 1 \). Prove using complete induction that for all \( n \geq 1 \), \( a_n = 2f_n - 1 \), where \( f_n \) denotes the \( n \)-th Fibonacci number. (We’ll see next week how we might come up with an expression like \( 2f_n - 1 \).)

2. Consider the following recursively defined set \( S \) of (some) binary strings.
   - \( \epsilon \) (the empty string “”) and 1 are in \( S \).
   - If \( w \) is a string in \( S \), then so is \( w0 \).
   - If \( w \) is a string in \( S \), then so is \( w01 \).

   (a) [2] Come up with a simple English description of the strings in \( S \). (Hint: try writing out all of the strings in \( S \) of length \( \leq 4 \) or 5. The property they share is quite simple.)

   (b) [6] Prove, using structural induction, that every string in \( S \) satisfies the description you gave in part (a).
Prove the converse: that every string with the property you identified in part (a) is actually in $S$.

Hint: use complete induction on the length of the string, with the predicate $P(n)$: every string of length $n$ that has “the property” is in $S$. Consider the different cases for the last character.

3. [2] Consider the following induction “proof” that everyone in the world has the same favourite colour.

Proof. We’ll prove that for all $n \geq 1$, $P(n)$: in every group $S$ of size $n$, each person in $S$ has the same favourite colour. For the base case, we let $n = 1$, and $S$ be a group of size 1. Then certainly everyone in $S$ has the same favourite colour – there is only one person!

Now let $k \geq 1$, and assume that for all groups $S$ of size $j$ with $1 \leq j \leq k$, everyone in $S$ has the same favourite colour. This is the induction hypothesis using complete induction. Now consider a group $S$ of size $k + 1$. Split up $S$ into two smaller groups $S_1$ and $S_2$ of size $\leq k$. By the induction hypothesis, everyone in $S_1$ has the same favourite colour – call it $c_1$ – and everyone in $S_2$ has the same favourite colour – call it $c_2$. Now pick one person $x \in S_1$ and one person $y \in S_2$, and consider the group $\{x, y\}$. By the induction hypothesis again, $x$ and $y$ must have the same favourite colour, and so $c_1 = c_2$. Therefore everyone in $S$ has the same favourite colour, and this concludes our proof.

Clearly, there must be something wrong with this proof, as not everyone in the world has the same favourite colour. Explain what is wrong.

Programming Question(s): purely for your benefit

Occasionally, we will suggest programming exercises for you to try to reinforce the concepts presented in lecture and on the problem set. These are not to be handed in, but we think they can be very beneficial to your own learning (not to mention valuable coding practice). Feel free to discuss these questions with your peers and the course staff throughout the term!

This challenge is open-ended and rather difficult. In goes without saying that you shouldn’t feel obligated to complete this within a week.

Write a program which generates all words in $S$ (defined in the questions above) with length less than or equal to $n$, where $n$ is some input parameter to your program.