CSC236 fall 2014, Assignment 1

Due October 3rd, 10 p.m.

The aim of this assignment is to allow you to explore some problems containing recursive components, and then use induction or well-ordering to solve those probkems.

There are 3 questions (be sure to look at page 2). You may work in groups of no more than three students, and you should produce a single solution in a PDF file named `a1.pdf`, submitted to MarkUs. You will receive 20% of the marks for any question (or pat of a question) that you either leave blank or for which you write "I cannot answer this."

1. We've shown for $m = 3$ or 4 and most natural numbers n, that $m^n \geq n^m$. It's tedious to repeat this proof for all the natural numbers $m > 1$, so use induction on n to prove:

$$\forall m \in \mathbb{N}, \exists k \in \mathbb{N}, \forall n \in \mathbb{N}, (m > 1 \wedge n \geq k) \Rightarrow m^n \geq n^m$$

   Hint #1: You are proving a claim with multiple quantifiers, but you only need induction for the innermost, $\forall n \in \mathbb{N}$.
   Hint #2: You may find it useful to notice that

$$(n+1)^m = \left(\frac{n+1}{n}\right)^m \times n^m = (1 + 1/n)^m \times n^m$$

2. Consider the following equation:

$$(\sqrt{5}+2)(\sqrt{5}-2) = 1$$

   You can complete parts (a) and (b) before you learn about the Principle of Well Ordering. Once you've learned the PWO, you can complete (c).

   (a) Re-write the equation until you have an equation with $\sqrt{5}$ on the left and a ratio of two expressions involving $\sqrt{5}$ on the right.
   Hint #1: Don't multiply out the bracketed expressions on the left — you'll lose the $\sqrt{5}$s.
   Hint #2: There are two similar ways to do this, yielding different expressions on the right.

   (b) Assume there are natural numbers m and n such that $\sqrt{5} = m/n$. Substitute m/n for $\sqrt{5}$ in your equation from (a). Simplify the ratio on the right hand side, to get a fraction of integers with the denominator a natural number less than n. If this doesn't work, go back to (a) and derive the other expression, then try that one.
   Show your substitution and simplification, and explain why the denominator is less than n.

   (c) Use the Principle of Well Ordering to derive a contradiction from the assumption in the previous part. What can you conclude?

3. Read over, and experiment with, the Python function **is_b_list**:

```
def is_b_list(x):
    """(object) -> bool

    Return whether x is a binary list.

    >>> is_b_list("b_list")
    False
    >>> is_b_list(0)
    True
    >>> is_b_list([0, 0])
    True
    >>> is_b_list([[0]])
    False
    """
    return (x == 0 or
            (isinstance(x, list) and len(x) == 2
             and all([is_b_list(y) for y in x])))
```

Define the size of a binary list as the number of left brackets in its Python representation, i.e. the total number of list objects in the binary list. So `0` has size 0 and `[0, 0]` has size 1.

You can complete part (a) before we finish covering examples of induction. The particular Inductive Principle and associated proof form you'll rely on in (b) has a special name: "complete" induction. If you work on part (b) before we cover that principle you should try to convince yourself by inductive reasoning. The principle will then match your reasoning or help you complete your reasoning.

(a) Experiment until you find a formula (probably recursive) that computes the number of different binary lists of size **s**. Notice that if you call your formula **C(s)**, then **C(0)** computes 1 and **C(1)** also computes 1.

(b) Use complete induction to prove that your **C(s)** correctly computes the number of different binary lists of size **s**