ICS 6D—Winter 2024 (Dillencourt)
Practice problems for Test 2

1. Consider the following theorem:

   Theorem: For any integer n such that $n \geq 40$, it is possible to make n cents of postage using only 5-cent or 11-cent stamps.

   Use strong induction to prove this theorem, by (1) stating and proving the base cases and (2) filling in the blanks in the Inductive Step as given below.

   Base Cases: (Write down the base cases, and prove that each base case is true.)

   Inductive Step: (For each blank, write what should go in the blank.)

   Let k be an integer with $k \geq$ ___(a)___. Assume that it is possible to make j cents worth of postage for any j in the range from ___(b)___ to ___(c)___, inclusive, and show that it is possible to make $k+1$ cents worth of postage.

   Since ___(d)___ $\geq$ ___(e)___, it falls in the range from 40 to k, inclusive. Therefore we know by the inductive hypothesis that it is possible to make ___(f)___ cents worth of postage, using only 5-cent and 11-cent stamps. Adding one more 5-cent stamp makes ___(g)___ cents worth of postage.

2. Consider a set of strings `S` over the alphabet {`a`,`b`} defined recursively as follows:

   - **Base Case:** $\lambda \in$ `S`.
   - **Recursive rules:** if `x` $\in$ `S` and `y` $\in$ `S`, then:
     - `aaxb` $\in$ `S` (Rule 1)
     - `xy` $\in$ `S` (Rule 2)

   For any string `x`, define A(`x`) to be the number of `a`'s in `x`, and B(`x`) to be the number of `b`'s in `x`.

   Consider the following theorem:

   > **Theorem:** For every string `x` $\in$ `S`, $A(x) = 2 \cdot B(x)$.

   Use structural induction to prove this theorem, by filling in the missing portions of the proof. (For each blank, write what should go in the blank.)

   Base Case:

   $A(\lambda) =$ ___(a)___.

   $2 \cdot B(\lambda) = 2 \cdot$ ___(b)___ $=$ ___(c)___.

   Since both sides are equal to ___(d)___, $A(\lambda) = 2 \cdot B(\lambda)$.

   Inductive step: Let `x` be a string in `S` that is created by one or more applications of a recursive rule. There are two cases.

   Case 1: Rule 1 was the last rule applied to get `x`. Then `x` $=$ ___(e)___ for some string `y` $\in S$, so $A(x) = A(y) +$ ___(f)___ and $B(x) = B(y) +$ ___(g)___. Using these last two equations and the inductive hypothesis, we see that:

   $$\begin{aligned} A(\mathtt{x}) &= A(\mathtt{y}) + \underline{\quad (h) \quad} \\ &= \underline{\quad (i) \quad} \quad \text{(By the Induction Hypothesis)} \\ &= 2 \cdot (\underline{\quad (j) \quad}) \\ &= 2 \cdot B(\mathtt{x}) \end{aligned}$$

   Case 2: Rule 2 was the last rule applied to get `x`. Then `x` $=$ ___(k)___ for strings `y` $\in S$ and `z` $\in S$, so $A(x) =$ ___(l)___ $+$ ___(m)___, and similarly $B(x) =$ ___(n)___. Using these last two equations and the inductive hypothesis, we see that:

   $$\begin{aligned} A(\mathtt{x}) &= \underline{\quad (o) \quad} \\ &= \underline{\quad (p) \quad} \quad \text{(By the Induction Hypothesis, applied twice)} \\ &= 2 \cdot (\underline{\quad (q) \quad}) \\ &= 2 \cdot B(\mathtt{x}) \end{aligned}$$

3. Consider the following problem:

> Give a recursive algorithm that takes as input a string s and returns the number of `a`'s in s. (For example, if the input is `aabcacb`, the algorithm should return 3.)

An algorithm that solves this problem, with a few missing portions, is shown in the box below. Fill in the missing portions. (For each blank, write what should go in the blank.)

```
a_count(s)

// Input:  A string s

// Output:  The number of occurrences of a in s.

If s = ___(a)___
    Return ___(b)___

Let c be the first character in s

Let t be the string s with the first character removed

k := ___(c)___     (Recursive call)

If c = ___(d)___
    k := ___(e)___

Return k
```