CSC236 – Problem Set 4

There are two components of this problem set. The preliminary question is not marked or submitted: it is there as a suggested exercise that you should do early to make sure that you're on track. The problem set itself is what you will submit for marks.

Get in the habit of starting work early – the less time you give yourself, the more stressed you'll find yourself each week!

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

The PDF file you submit must be typed, scanned handwritten submissions will not be marked.

Preliminary: Not Marked

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!

Here is a recursively-defined function.

$$T(n) = \begin{cases} 5, & \text{if } n = 1 \\ T(n-1) + 3, & \text{if } n > 1 \end{cases}$$

Carry out the five steps for repeated substitution to prove a closed-form for this recursive definition. Split up your five steps so that we know where each step begins and ends.

Problem Set: due October 14, 2016 22:00, required filename: ps4sol.pdf

Answer each question completely, always justifying your claims and reasoning. Your solution will be graded not only on correctness, but also on clarity.

Answers that are technically correct 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.

1. **[7]** Here is a recursively-defined function:

$$f(n) = \begin{cases} 5, & \text{if } n = 0 \\ 8, & \text{if } n = 1 \\ 5f(n-1) - 4f(n-2), & \text{if } n \geq 2 \end{cases}$$

   Carry out the five steps for repeated substitution to prove a closed-form for this recursive definition. Split up your five steps in your submission so that we know where each step begins and ends.

   Hints:

   (a) After k substitutions, $f(n)$ should be in terms of $f(n-k)$ and $f(n-k-1)$.

   (b) In Step 2, the patterns in the coefficients before $f(n-k)$ and $f(n-k-1)$ might not be obvious and may require some effort to guess. What you can do is to observe the recursive relation between the coefficients when k is increased by 1, and perform another find-closed-form operation for the coefficients. You are not required to show this guessing process in your submission, but it is a good practice. The guessed formula should have 4^k in it.

   (c) In Step 3 and 4, you will solve a single k that covers both base cases, so you only need to do Step 3 and 4 *once*.

2. **[5]** Consider the following pseudo-code of an algorithm `mystery`:

```
def mystery(lst):
    if len(lst) <= 1:
        return
    if lst[0] > lst[-1]:
        lst[0], lst[-1] = lst[-1], lst[0]
    if len(lst) >= 3:
        split = len(lst) // 3
        mystery(lst[0..len(lst) - split - 1])
        mystery(lst[split..len(lst) - 1])
        mystery(lst[0..len(lst) - split - 1])
```

Find a recursive definition for $T(n)$, the worst-case runtime of `mystery`. State your recursive definition and justify clearly why it is correct. You don't need to find its closed form.

Notes:

(a) Assume that list-slicing is performed in constant time, as discussed in lecture as well as on pages 32-33 of the course notes.

(b) The "dot-dot notation" (`..`) is defined the same way as we discussed in the lecture. Refer to Week 5 lecture slides for more details.

(c) Like what we did in the lecture examples, use variables such as c and d to express a constant amount of runtime.