CS 61A Summer 2009 Homework 2

Topic: Abstract Data

Lectures: Monday 6/29, Tuesday 6/30, Wednesday 7/1, Thursday 7/2

Reading: Abelson & Sussman, Section 1.1 (pages 1–31), Section 1.3

1. Abelson and Sussman: 1.16, 1.37, 1.38 (1.16 is a bit tricky but very important)

2. (Graded by Correctness) We would like a procedure that takes a word and sentence and returns how many times the input word appears in the sentence. First, define a procedure `(count-occur-r wd sent)` that uses a recursive process to solve this problem. Then define a procedure `(count-occur-i wd sent)` that uses an iterative process to solve this problem.

3. Consider the following definitions:

```
(define (1+ n) (+ 1 n))

(define (many+ n)
   ((repeat 1+ n) n)

(define (foo n)
   ((repeat many+ n) n))

(define (bar n)
   (foo (foo n)))
```

Where we use the `repeat` procedure from lecture and in SICP. What are the asymptotic runtimes of the procedures `1+, n+, foo, bar` in term of input `n`?

4. Consider the following procedure `all-pair-sums` that takes two sentences of numbers and returns a sentence of all numbers formed by adding one element from the first input sentence and one element from the second input sentence.

```
(define (all-pair-sum sent1 sent2)
  (if (empty? sent2) '()
      (se (every (lambda (num) (+ (first sent2) num)) sent1)
          (all-pair-sum sent1 (butfirst sent2)))))
```

Suppose both both input sentences have similiar length n, what is the asymptotic runtime of the procedure `all-pair-sums` in term of n? Is it possible to be smarter and write an `all-pair-sum` procedure that runs in time linear to n? Explain your answers concisely.

5. Abelson and Sussman, exercises 2.17, 2.20, 2.22, 2.24, 2.26 (these should be relatively short, no need to turn in the box-pointer diagram)

6. Abelson and Sussman, exercise 2.29 (this is a long but very worthwhile question)

7. Abelson and Sussman, exercises 2.30, 2.32

8. Deep-lists are useful to represent hierachical data, data that naturally has levels. We can in fact still use a flat-list to represent hierachical data by using special prefixes. For example, to represent the following list of organisms categorized by similarity:

```
( ( (snail slug) (moose deer) ) (oak willow) )
```

We could instead add prefixes and then put them into one sentence:

```
1A1.snail
1A2.slug
1B1.moose
1B2.deer
2A.oak
2B.willow

(1A1.snail 1A2.slug 1B1.moose 1B2.deer 2A.oak 2B.willow)
```

We need to alternate between numbers and letters because of multi-digit numbers. We use notation `1AA.duck` if duck is the 27th item in that category. This scheme is developed by Douglas Engelbart (inventor of the mouse) and presented with many other innovations in a live demonstration in 1968. That highly influential presentation is now known as "The Mother of All Demos".

Your job is to write a procedure `(engelbart ls)` that takes a deep-list of words and converts it to a flat-list via the Engelbart notation. This is a more involved question and you should think carefully about how to design a short understandable program. For example, you could think about what abstract data types and helper procedures might help you.

Extra for experts:

A Quine is a program that prints itself. As an example, "write down this sentence" instructs the reader to write down those exact words on the paper. However, there is no self-referring concept of "this program" for a computer program.

Remember, the return value of your program must be the code of the program exactly. **Do not omit any letter (even quotes) and do not include any extra letters.**

Computer viruses are basically Quines since they write down the exact code of their own program when they infect a new computer. In fact, living beings are all Quines if you think about it; the genetic information in your DNA tells your body, among other things, how to build that very DNA.

Quines allow computers to reproduce. Quines make it possible to build a robot who is programmed with the instruction to build an exact copy of itself. With the existence of Quines, one really has to wonder what exactly is the difference between a machine a living being. If we build reproducing machines and add some mechanism for evolution, could we create a new type of life?

Anyways, hope that is enough motivation for you to try this very interesting problem (and you actually know enough to do it!!)