Homework 1

SNU 4910.210, Fall 2014

Chung-Kil Hur

due: 9/21(Sun), 24:00

The objectives of this homework are:

- to learn how to combine basic components of programming; and
- to learn how to use recursion.

Exercise 1 "Euclid gcd"

Input: two non-negative integers n and m.

Define a function `gcd` that evaluates to the greatest common divisor (`gcd` n m). Hint: (Euclid method) do recursion using the following property: gcd of n and 0 is n; and gcd of n and m ($n \geq m > 0$) is equal to gcd of $n - m$ and m. □

Exercise 2 "Turing 2"

Input: an integer n.

In 1936, Alan Turing introduced an abstract notion of computer, called Turing Machine, in his seminal paper[1]. The second example in the paper is a program that outputs $001011011101111\cdots$. Write a function `t2` such that (`t2` n) prints out $0010110111\cdots0\underbrace{1\cdots1}_{|n|}$. □

Exercise 3 "Yang Hui 3"

Input: a non-negative integer n

[1] "On Computable Numbers, with an Application to Eintscheidungsproblem", *Proceedings of the London Mathematical Society, ser.2, vol.42, pp.230-265., 1936.*

In China, Yang Hui already invented and used Pascal's triangle in 13th century.

$$
\begin{array}{ccccccc}
 & & & 1 & & & \\
 & & 1 & & 1 & & \\
 & 1 & & 2 & & 1 & \\
1 & & 3 & & 3 & & 1 \\
 & & & \vdots & & &
\end{array}
$$

Write a function `yanghui` that prints out Pascal's triangle up to the n'th row. For example, `(yanghui` n`)` prints out $1111211331\cdots\underbrace{1\cdots1}_{n\text{'th row}}$; and `(yanghui` 0`)` prints out nothing. □

Exercise 4 "zipper"

Input: two lists of integers

Write a function `zipper` that zips the given two lists. For example, `(zipper '(1 2 3 4) '(5 6))` evaluates to `(1 5 2 6 3 4)`; `(zipper '(1 2) '(3 4 5 6))` evaluates to `(1 3 2 4 5 6)`; and `(zipper '() '(1 2 3))` evaluates to `(1 2 3)`. □

Exercise 5 "zipperN"

Input: a list of lists of integers

Write a function `zipperN` that zips the given list of lists in order. For example, `(zipperN '((1 2 3) (4) (9 10 11 12)))` evaluates to `(1 4 9 2 10 3 11 12)`. □

Exercise 6 "Crazy-k"

Numbers in base-k ($k > 1$) are usually represented as follows:

$$d_0 \cdots d_n$$

where

$$\forall d_i \in \{0, \cdots, k-1\}.$$

and "$d_0 \cdots d_n$" denotes the integer

$$d_0 \times k^0 + \cdots + d_n \times k^n \ .$$

Let us define "crazy-k" as follows by slightly extending "base-k". Numbers in crazy-k ($k > 1$) are represented as follows:

$$d_0 \cdots d_n$$

where

$$\forall d_i \in \{1-k, \cdots, 0, \cdots, k-1\}.$$

and "$d_0 \cdots d_n$" denotes the integer

$$d_0 \times k^0 + \cdots + d_n \times k^n \ .$$

For example, consider crazy-2 with $\{-1, 0, 1\}$ as digits. Suppose that `0`, `+` and `-` represent 0, 1 and -1 respectively. Then, `+`, `+0+`, `+-` and `+-0-` denote 1, 5, -1 and -9 respectively.

We can inductively define the set N of numbers in crazy-2 as follows:

$$\begin{array}{rcl} N & ::= & \mathtt{0} \\ & | & \mathtt{+} \\ & | & \mathtt{-} \\ & | & \mathtt{0}N \\ & | & \mathtt{+}N \\ & | & \mathtt{-}N \end{array}$$

In Scheme, we can represent the set N using list, say $\underline{N}$, as follows:

$$\begin{array}{rcl} \underline{\mathtt{0}} & = & \mathtt{'z} \\ \underline{\mathtt{+}} & = & \mathtt{'p} \\ \underline{\mathtt{-}} & = & \mathtt{'n} \\ \underline{\mathtt{0}N} & = & \mathtt{(cons\ 'z\ }\underline{N}\mathtt{)} \\ \underline{\mathtt{+}N} & = & \mathtt{(cons\ 'p\ }\underline{N}\mathtt{)} \\ \underline{\mathtt{-}N} & = & \mathtt{(cons\ 'n\ }\underline{N}\mathtt{)} \end{array}$$

For instance, `0+-` is expressed as

```
(cons 'z (cons 'p 'n))
```

because

$$\begin{array}{rcl} \underline{\mathtt{0+-}} & = & \mathtt{(cons\ 'z\ }\underline{\mathtt{+-}}\mathtt{)} \\ & = & \mathtt{(cons\ 'z\ (cons\ 'p\ }\underline{\mathtt{-}}\mathtt{))} \\ & = & \mathtt{(cons\ 'z\ (cons\ 'p\ 'n)).} \end{array}$$

Now, define a function `crazy2val` that takes a number n in crazy-2 (represented as above) and evaluates to the integer that the number n denotes.

$$\mathtt{crazy2val} : \text{Crazy-2} \rightarrow \text{Integer}.$$

For example, `(crazy2val '(z p . n))` evaluates to -2. □

Exercise 7 "Addition in Crazy-2"

Define a function `crazy2add` that takes two numbers in crazy-2 and evaluates to their sum in crazy-2.

$$\texttt{crazy2add} : \text{Crazy-2} * \text{Crazy-2} \to \text{Crazy-2}.$$

`crazy2add` should satisfy the following properties:

- For any z and z' in crazy-2,

  $$\texttt{(crazy2val (crazy2add } z\ z'\texttt{))} = \texttt{(crazy2val } z\texttt{)} + \texttt{(crazy2val } z'\texttt{)}.$$

- `crazy2add` should be defined recursively. Note that it is not allowed to convert numbers in crazy-2 into integers, add them as integers, and revert the sum back into crazy-2.

  You can add two numbers $d_0 \cdots d_n$ and $d'_0 \cdots d'_m$ in crazy-2 by basically adding d_i and d'_i for each i. However, you should also consider the carry c_i that is transferred from the previous calculation at $i-1$'th column. Thus, in fact, you should calculate $d_i + d'_i + c_i$ for each i.

□