CS 2322 Extra Credit Laboratory Problem Set 4

October 4, 2011
Due October 18, 2011

These problems are to be done on an individual basis following the Trinity University Academic Integrity Policy or Trinity University Honor Code.

Academic Integrity and Honor Code

All students are covered by a policy that prohibits dishonesty in academic work. The Academic Integrity Policy (AIP) covers all students who entered Trinity before the Fall of 2004. The Academic Honor Code covers all those who entered the Fall of 2004 or later. The Integrity Policy and the Code share many features: each asserts that the academic community is based on honesty and trust; each contains the same violations; each provides for a procedure to determine if a violation has occurred and what the punishment will be; each provides for an appeal process. The main difference is that the faculty implements the AIP while the Honor Code is implemented by the Academic Honor Council. Under the Academic Integrity Policy, the faculty member determines whether a violation has occurred as well as the punishment for the violation (if any) within certain guidelines. Under the Honor Code, a faculty member will (or a student may) report an alleged violation to the Academic Honor Council. It is the task of the Council to investigate, adjudicate, and assign a punishment within certain guidelines if a violation has been verified. Students who are under the Honor Code are required to pledge all written work that is submitted for a grade: On my honor, I have neither given nor received any unauthorized assistance on this work and heir signature. The pledge may be abbreviated pledged with a signature.

Laboratory problems should be submitted electronically (e-mail to cs2322@cs.trinity.edu) on or before the due date and should contain a problem write-up, source code to any programs and data sets used in solving the problem. The submitted files should be ASCII text files having Unix end-of-line characters (please convert all Windows and Mac text files to Unix format–I have found that Emacs seems to do a reasonable job of such conversions). If several files need to be submitted, put them in a directory having name *your-last-name-problem-set-number* and create a tar archive of this file system and attach it to your e-mail problem submission.

Searching a Database

Suppose that `db` is a database having the following structure:

- A database is a list of boxed items called records.
- A record is a boxed list of J items which are considered to be atomic (will not be further subdivided even though it might be possible to so do).
- Each record (a boxed list of J items) contains a key which is defined to be the contents (open) of the first item in a record.
- It is assummed that each record consists of two or more items, a key and one or more items.

Define a dyad `locate` which will return a one record database consisting of the first record having a key which matches the given key. The left argument of `locate` should be the database and the right argument should be the search key. If the database contains no records with a matching key, `locate` should return an empty database.

Searching for all Records Matching a Key

Define a dyad `locate_all` which will return the database (possibly empty) of all records matching a given key. The left argument of `locate_all` should be the database and the right argument should be the search key. If the database contains no records with a matching key, `locate_all` should return an empty database.

Discussion

During the discussion of Laboratory Problem 4. we developed the idea of using an adverb

```
key =: 1 : '> @ (m&{) @ >'
```

to be used for searches where the *key* field is not allways the first item in a record. For example for the database `db`

```
   > db
+-+-+-+-+-+
|1|2|0|4|3|
+-+-+-+-+-+
|0|1|2|3|4|
+-+-+-+-+-+
|0|4|1|3|2|
+-+-+-+-+-+
|3|1|4|0|2|
+-+-+-+-+-+
|0|2|1|4|3|
+-+-+-+-+-+

   0 key db
1 0 0 3 0
   0 = 0 key db
0 1 1 0 1
```

which provides useful functionality, but when the key items are not J atoms, such as is the case for the `people` database:

```
   > people
+-------+-------+-------------------------+
|howland|jack   |computer science professor|
+-------+-------+-------------------------+
|clinton|bill   |president usa            |
+-------+-------+-------------------------+
|clinton|hillary|president usa            |
+-------+-------+-------------------------+
|perot  |ross   |wanted to be president   |
+-------+-------+-------------------------+
|bush   |george |used to be president     |
+-------+-------+-------------------------+
   0 key people
```

```
howland
clinton
clinton
perot
bush
   'perot' (-:"1) 0 key people
0 0 0 0 0
```

The `key` adverb fails because applying `0 key` produces a function, which when applied, produces a result where shorter keys are padded on the right with the appropriate fill character for the type (space in the above example).

Extra Credit Problem

Define a conjuction, `key` which has a key-field number as left argument and key-search value as right argument which produces a function which embeds the matching on a key by key basis so that:

```
   0 key 'perot' "0 people
0 0 0 1 0
   0 key 0 "0 db
0 1 1 0 1
```

Problem Set 4 Solution [HTML] [PS] [PDF]