WEEK 07

1. Preparation for Assignment

If, and *only if* you can truthfully assert the truthfulness of each statement below are you ready to start the exercises.

1.1. Reading Comprehension Self-Check.

- I know that **input enhancement** is the idea of preprocessing some or all of a problem's input, and storing the additional information obtained to accelerate solving the problem afterward.
- I know why it is **false** to say that the two principal resources of time and space compete with each other in **all** design situations.
- We know why sorting by distribution counting is more efficient than quicksort.
- I know why it is **false** to say that data compression is a typical space-time tradeoff.
- I know that **prestructuring** creates structures that allow faster and/or more flexible access to data.
- I know why it is **false** to say that these pre-structured structures typically use less space than otherwise required.
- I know that hashing enables, on average, constant-time searching, insertion, and deletion.

1.2. **Memory Self-Check.** Which hash function is better for a key that is a character string? (The parameter m is the modulus, the size of the hash table.)

- hash-function-1, or
- hash-function-2?

```
1 BEGIN_SRC emacs-lisp
2   (require 'cl)
3
4   (defun hash-function-1 (key m)
5     (mod (loop for c across key sum c) m))
6 END_SRC
7
8 BEGIN_SRC emacs-lisp
9   (require 'cl)
10
11  (defun hash-function-2 (key m)
```

Date: November 7, 2018.

```
12      (let ((h 0))
13        (loop for c across key do (setq h (+ (* h 31) c)))
14        (mod h m)))
15 END_SRC
```

2. Week 07 Exercises

2.1. **Exercise 1 on page 274.**

2.2. **Exercise 2 on page 274.**

2.3. **Exercise 3 on page 275.**

2.4. **Exercise 4 on page 275.**

2.5. **Exercise 3 on page 279.**

2.6. **Exercise 6 on page 279.**

3. Week 07 Problems

3.1. **Not in the Book.** The birthday problem is implemented in the following code. Study it, then answer the question, why does this code not give 23 as the answer?

```
1 BEGIN_SRC emacs-lisp
2   (require 'cl)
3
4   (defun birthday-problem (&optional max-num)
5     (if (null max-num) (setq max-num 10000))
6     (let ((num-days-in-year 365)
7           (grand-sum 0)
8           (n 0)
9           A counter)
10      (loop repeat max-num
11            do (setq A (make-vector num-days-in-year 0)
12                     counter 0)
13            (loop until (= 2 (elt A n))
14                  do (setq n (random num-days-in-year))
15                  (incf (elt A n))
16                  (incf counter))
17            (incf grand-sum counter))
18      (ceiling (/ grand-sum (float max-num)))))
19 END_SRC
```