# 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 1, 2018.

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

## 2. Week 07 Exercises

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

A.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 56 | | | | | | | 30 | 20 |
| | | | | | | | | | 19 | 75 |
| | | | | | | | | | | 31 |

B. find the largest number of key comparisons in a successful search in this table. Searching for K = 31, the number is 3.

C. 1.7 or 1.66 rounded up

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

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 30 | | |
| | | | | | | | | | 30 | 20 | |
| | | 56 | | | | | | | 30 | 20 | |
| | | 56 | | | | | | | 30 | 20 | 75 |
| | 31 | 56 | | | | | | | 30 | 20 | 75 |
| | 31 | 56 | 19 | | | | | | 30 | 20 | 75 |

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

2.4. **Exercise 4 on page 275.** The probability of all n keys to be hashed to a particular address is equal to (1/m)n. Since there are m different addresses, the answer is

$(\frac{1}{m})^n$ * m = $1/(m^{n-1nil})$ .

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?

```
BEGIN_SRC emacs-lisp
  (require 'cl)

  (defun birthday-problem (&optional max-num)
    (if (null max-num) (setq max-num 10000))
    (let ((num-days-in-year 365)
          (grand-sum 0)
          (n 0)
          A counter)
      (loop repeat max-num
            do (setq A (make-vector num-days-in-year 0)
                     counter 0)
            (loop until (= 2 (elt A n))
                  do (setq n (random num-days-in-year))
                  (incf (elt A n))
                  (incf counter))
            (incf grand-sum counter))
      (ceiling (/ grand-sum (float max-num)))))
END_SRC
```