Principles of Programming, Spring 2006
Practice 8
Modeling with Mutable Data

Daejun Park, Heejong Lee
Programming Research Lab.@SNU

May 4, 2006

1. A *queue* is a sequence in which items are inserted at one end (called the *rear* of the queue) and deleted from the other end (the *front*). A *ordered-queue* is, however, a sequence in which items are inserted at one end and deleted from the least element, where items are ordered numerically. For each implementation of *queue* and *ordered-queue*, we need the following procedures. Define all the procedures.

`make-queue`	:	$unit \to queue$
`insert-queue!`	:	$queue \times number \to queue$
`delete-queue!`	:	$queue \to queue$
`make-ordered-queue`	:	$unit \to queue$
`insert-ordered-queue!`	:	$queue \times number \to queue$
`delete-ordered-queue!`	:	$queue \to queue$

For examples,

```
(define q (make-queue))
(insert-queue! q 2)
(insert-queue! q 1)
(delete-queue! q)
; 2
(delete-queue! q)
; 1
(define q (make-ordered-queue))
(insert-ordered-queue! q 2)
(insert-ordered-queue! q 1)
(delete-ordered-queue! q)
; 1
(delete-ordered-queue! q)
; 2
```

2. You have implemented *dictionary* in Exercise 5 of Homework 2. The *dictionary* was not a mutable data structure. That is, even though you changed the *dictionary* using `dictInsert` or `dictDelete`, the original *dictionary* was not changed, but a new *dictionary* was created. From now on, implement a mutable *dictionary* data structure. Define the following procedures.

`make-dict`	:	$unit \to dict$
`lookup`	:	$key \times dict \to value$
`insert!`	:	$key \times value \times dict \to unit$
`delete!`	:	$key \times dict \to unit$

For examples,

```
(define d (make-dict))
(insert! 1 10 d)
(insert! 2 20 d)
(lookup 1 d)
; 10
(insert! 1 30 d)
(lookup 1 d)
; 30
(delete! 2 d)
(lookup 2 d)
; #f
```