CS135 Tutorial 05

Recursion with more lists and association lists

Goals

- Get more comfortable with recursing on 2 lists at the same time, consuming association lists
- Practice some tracing

CS135 Search Data Definition

```
;; A doc-list (DL) is one of:
;; * empty
;; * (cons Str DL)
;; Requires: each doc (i.e. Str) only occurs once in the doc-list
;; the doc-list is in lexicographic order

;; An Inverted List (IL) is one of:
;; * empty
;; * (cons (list Str DL) IL)
;; Requires: each key (i.e. Str) only occurs once in the IL.
;; the keys occur in lexicographic order in the IL.
```

CS135 Search: both

Create a function `both` which consumes two `DL`s and produces a doc-list (`DL`) that occur in both `DL`s. For example,

```
(both (list "b.txt") (list "b.txt" "c.txt"))
=> (list "b.txt")
```

Hint: We can take advantage that doc-lists are sorted

We can also use the following predicates:

- `string<?`
- `string<=?`
- `string=?`
- `string>?`
- `string>=?`

CS135 Search: both (trace)

```
(both (list "a.txt" "b.txt" "c.txt") (list "b.txt" "c.txt" "d.txt"))
=> (both (list "b.txt" "c.txt") (list "b.txt" "c.txt" "d.txt"))
=> (cons "b.txt" (both (list "c.txt") (list "c.txt" "d.txt"))
=> (cons "b.txt" (cons "c.txt" (both (list) (list "d.txt"))
=> (cons "b.txt" (cons "c.txt" empty)
= (list "b.txt" "c.txt")
```

CS135 Search: exclude

Create a function `exclude` which consumes two `DL`s and produces a doc-list (`DL`) that occur in the first `DL` but not the second one. For example,

```
(exclude (list "b.txt" "c.txt") (list "b.txt"))
=> (list "c.txt")
```

CS135 Search: both -> exclude

```
(define (both doc-lst1 doc-lst2)
  (cond [(or (empty? doc-lst1) (empty? doc-lst2)) empty]
        [(string=? (first doc-lst1) (first doc-lst2))
         (cons (first doc-lst1) (both (rest doc-lst1) (rest doc-lst2)))]
        [(string<? (first doc-lst1) (first doc-lst2))
         (both (rest doc-lst1) doc-lst2)]
        [else (both doc-lst1 (rest doc-lst2))]))
```

CS135 Search: both -> exclude

```
(define (exclude doc-lst1 doc-lst2)
  (cond [(empty? doc-lst1) empty]
        [(empty? doc-lst2) doc-lst1]
        [(string=? (first doc-lst1) (first doc-lst2))
         (cons (first doc-lst1) (both (rest doc-lst1) (rest doc-lst2)))]
        [(string<? (first doc-lst1) (first doc-lst2))
         (both (rest doc-lst1) doc-lst2)]
        [else (both doc-lst1 (rest doc-lst2))]))
```

CS135 Search: both -> exclude

```
(define (exclude doc-lst1 doc-lst2)
  (cond [(empty? doc-lst1) empty]
        [(empty? doc-lst2) doc-lst1]
        [(string=? (first doc-lst1) (first doc-lst2))
         ~~(cons (first doc-lst1)~~ (exclude (rest doc-lst1) (rest doc-lst2)))]
        [(string<? (first doc-lst1) (first doc-lst2))
         (both (rest doc-lst1) doc-lst2)]
        [else (both doc-lst1 (rest doc-lst2))]))
```

CS135 Search: both -> exclude

```
(define (exclude doc-lst1 doc-lst2)
  (cond [(empty? doc-lst1) empty]
        [(empty? doc-lst2) doc-lst1]
        [(string=? (first doc-lst1) (first doc-lst2))
         (exclude (rest doc-lst1) (rest doc-lst2))]
        [(string<? (first doc-lst1) (first doc-lst2))
         (cons (first doc-lst1) (exclude (rest doc-lst1) doc-lst2))]
        [else (both doc-lst1 (rest doc-lst2))]))
```

CS135 Search: both -> exclude

```
(define (exclude doc-lst1 doc-lst2)
  (cond [(empty? doc-lst1) empty]
        [(empty? doc-lst2) doc-lst1]
        [(string=? (first doc-lst1) (first doc-lst2))
         (exclude (rest doc-lst1) (rest doc-lst2))]
        [(string<? (first doc-lst1) (first doc-lst2))
         (cons (first doc-lst1) (exclude (rest doc-lst1) doc-lst2))]
        [else (exclude doc-lst1 (rest doc-lst2))]))
```

Done!

CS135 Search: doc-retrieve

Create a function `(keys-retrieve doc an-il)` which consumes a `Str` and an `IL` and produces a `(listof Str)` with lexicographic ordering. The values in the produced list are the keys from `an-il` whose doc-lists contain `doc`. If `doc` is not contained in the doc-list associated with any keys in `an-il`, then `keys-retrieve` produces `empty`.

CS135 Search: search

Create a function `search` which consumes a `Sym`, two `Strs` and an `IL`. It produces a doc-list (`DL`). The arguments for search will always be in one of two possible formats:

- `(search 'both str1 str2 an-il)` which, given two keys `str1` and `str2` from `an-il`, produces a doc-list (DL) containing the documents that are present in both of the keys' associated doc-lists.
- `(search 'exclude str1 str2 an-il)` which, given two keys `str1` and `str2` from an-il, produces a doc-list (DL) containing the documents that are present in the doc-list associated with the key `str1`, but not the key `str2`.