CS 135 — L07: Producing & Transforming Lists

Luke Lu • 2025-11-11

Website: student.cs.uwaterloo.ca/~cs135

.

🐳 **Info** — In these notes we lean on the **design recipe for lists** and show patterns with small reusable templates.

Filtering (select by predicate)

```
;; keep only even numbers
(define (keep-even lst)
  (cond [(empty? lst) empty]
        [(odd? (first lst)) (keep-even (rest lst))]
        [else (cons (first lst) (keep-even (rest lst)))]))
```

💡 **Tip** — **Template** Replace `odd?` with any predicate to build your own filter.

Generic filter:

```
(define (filter p? lst)
  (cond [(empty? lst) empty]
        [(p? (first lst)) (cons (first lst) (filter p? (rest lst)))]
        [else (filter p? (rest lst))]))
```

Mapping (transform each element)

```
(define (map f lst)
  (cond [(empty? lst) empty]
        [else (cons (f (first lst)) (map f (rest lst)))]))
```

Examples:

```
(map add1 '(1 2 3))             ;; -> (2 3 4)
(map (lambda (s) (if (symbol=? s 'apple) 'orange s)) '(pear apple pear))
```

Ordered Lists & Insertion

Check non-decreasing order:

```
(define (increasing? lst)
  (cond [(or (empty? lst) (empty? (rest lst))) true]
        [(> (first lst) (first (rest lst))) false]
        [else (increasing? (rest lst))]))
```

Insert while maintaining order:

```
(define (insert n lst)
  (cond [(empty? lst) (list n)]
        [(< n (first lst)) (cons n lst)]
        [else (cons (first lst) (insert n (rest lst)))]))
```

Last Element (non-empty)

```
(define (last lst)
  (cond [(empty? (rest lst)) (first lst)]
        [else (last (rest lst))]))
```

Question	Answer
Base case?	`empty` -> produce the neutral element (e.g., `empty`, `0`, etc.)
What to do with first?	Decide to keep/change/drop `(first lst)`
How to combine?	Use `cons`, `+`, etc. with the recursive result