CS 135

Midterm Help Session (with your ISAs)

Fall 2024

Module 2: Functions

↳ Stepping Rules for:

1. Built-in Functions
2. User Defined Functions
3. Constants

Module 2: Functions

What are the **first**, **second**, and **final** substitution steps?

racket

```
(define (double-sus x) (* x 2))

(double-sus (+ 3 4))
```

Module 3: Simple Data

1. Short Circuit Evaluation: (and ...), (or ...)
2. Stepping Rules for cond

Module 3: Simple Data

What are the **first**, **second**, and **final** substitution steps?

racket

```
(or (and (= 2 3) true) false)
```

Module 3: Simple Data

How could this be rewritten as a **Boolean** expression?

</> racket

```
;; red-sus?: Sym Sym -> Bool
(define (red-sus? task1 task2)
  (cond
    [(symbol=? task1 'wires) (symbol=? task2 'cardswipe)]
    [(symbol=? task1 'power) (symbol=? task2 'oxygen)]
    [(symbol=? task1 'wires) (symbol=? task2 'asteroids)]
    [else false]))
```

Module 4: Design Recipe

1. Design Recipe Components
2. Testing Your Code

Module 5: Structures

```
(define-struct person (name age id))
;; A person is a (make-person Str Nat Nat)
```

When defining a new structure in Racket, the following functions are automatically created:

1. **Constructor**: make-person
2. **Selectors**: person-name, person-age, person-id
3. **Predicate**: person?

The newly created structure data type can be used in contracts.

Module 6: Lists

What :

1. Should the function produce in the **base case**?
2. Should the function do to the **first element** of a **non-empty list**?
3. Should applying the function to the **rest of the list** produce?

How :

1. Should the function **combine** What.2 and What.3 to produce the answer?

Module 6: Lists

What are the **first**, **second**, and **final** substitution steps?

racket

```
(define (dup x) (list x x))

(rest (rest (dup (list 'blue 'red))))
```

Module 6: Lists

Insertion Sort

How can I alter the insertion-sort functions to change the sorting order?

</> racket

```
(define (sort lon)
  (cond [(empty? lon) empty]
        [else (insert (first lon) (sort (rest lon)))]))

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

Module 7: Natural Numbers

1. Data Definitions and Templates
   - How can I write a template according to the data definition?
2. Natural Numbers
3. Count-**DOWN** and Count-**UP**

Module 8: More Lists

1. Dictionaries
2. Association Lists
    - What are the **similarities** and **differences** between the two?

From your ISAs:

Good Luck on the Midterm!