Vantage 140 CPSC 110

Modeling with Cross Product Tables

Week 7

Lesson Plan:
Creating a Cross Product table as a model of a solution when dealing with a function consuming two One-Of types.

Intended Learning Outcomes:
- Develop understanding of how to construct the axes of a cross product table from type comments
- Develop understanding of how to fill in the cells of a cross product table by working through examples of the problem
- Develop understanding of how to simplify the cross-product table based on cell contents
- Develop understanding of how create check-expects, template and function body based on cross product table
- Develop strategies for solving a problem when it is not immediately clear what the solution is at a given step.

Required Materials:
1) 2-OneOf-problems.rkt
2) Flipchart paper + markers

Lesson Procedure		
	Resource(s)	**Time**
Main Task 1 – Solve problem in groups Put students into groups of 3. -Have them develop sig, purp, cross-product table for problem -Encourage/force them to write their strategies for solving the problem on their flipchart paper	-2-OneOf-problem-1.rkt -flipchart paper -marker for each group	15 mins

Main Task 1 – Go over solution -get groups to share strategies from flipchart -pick a group to demo solution -ask the following questions, which they should answer based on the cross-product table 1) ask how many C-Es needed and what cases are they testing? 2) ask how many cases in the cond and what are the questions in each case?	2-OneOf-problem-1.rkt -flipchart paper -marker for each group	15 mins
Main Task 2 – Solve problem in groups -Have groups develop sig, purp, cross-product table for problem -Encourage/force them to look to their strategies when they get stuck AND to write new strategies if they come up with some -take pictures of the flipchart paper + solution for each group at the end	-2-OneOf-problem-2.rkt -flipchart paper + marker for each group	25 mins

```
;; PROBLEM 1


;; ListOfNumber is one of:
;; - empty
;; - (cons Number ListOfNumber)
;; interp. a list of numbers
(define LON0 empty)
(define LON1 (cons 1 (cons 9.3 empty)))

(define (fn-for-lon lon)
  (cond [(empty? lon) (...)]
        [else
         (... (first lon)
              (fn-for-lon (rest lon)))]))



; For the following problem, provide the signature, purpose and cross-product table.
;
; While your group is filling in the cross-product table,
; add any new answers to the following questions on your flipchart paper.
;
; What did you do to help you understand the problem?
; What did you do when you didn't instantly know what to write in the solution?
;
;
; PROBLEM:
; Design a function that consumes two lists of numbers and produces true if
; the every element in the second list is strictly smaller than the corresponding
; element in the first list. You can assume that the two lists are of equal length.
;
```

```
;; PROBLEM 2
 (define-struct node (key val l r))
;; BST is one of:
;; - false
;; - (make-node Integer String BST BST)
;; interp.
;; - false means no BST (empty BST)
;; - key is the node key
;; - val is the node value
;; - l and r and left and right subtrees
;; INVARIANT for a given node
;; key is > than all keys in the left subtree
;; key is < than all keys in the right subtree
;; the same key never appears twice in the tree

(define BST0 false)
(define BST1 (make-node 1 "abc" false false))
(define BST7 (make-node 7 "ruf" false false))
(define BST4 (make-node 4 "dcj" false BST7))
(define BST3 (make-node 3 "ilk" BST1 BST4))
(define BST42
  (make-node 42 "ily"
          (make-node 27 "wit" (make-node 14 "olp" false false) false)
          (make-node 50 "dug" false false)))
(define BST10
  (make-node 10 "why" BST3 BST42))
#;
(define (fn-for-bst t)
  (cond [(false? t) (...)]
        [else
         (... (node-key t)              ; Integer
              (node-val t)              ; String
              (fn-for-bst (node-l t))     ; BST
              (fn-for-bst (node-r t)))])) ; BST

;; ListOfInteger is one of:
;; - empty
;; - (cons Integer ListOfInteger)
;; interp. a list of integers
(define LOI0 empty)
(define LOI1 (cons 1 (cons 93 empty)))

(define (fn-for-loi loi)
  (cond [(empty? loi) (...)]
        [else
         (... (first loi)
              (rest loi))]))
```

; For the following problem, provide the signature, purpose and cross-product table.
;
; While your group is filling in the cross-product table,
; add any new answers to the following questions on your flipchart paper.
;
; What did you do to help you understand the problem?
; What did you do when you didn't instantly know what to write in the solution?
;
;
; PROBLEM:
; Design a function that consumes a list of integers and a BST and produces true
; if the integers in the list are consecutive keys of t, starting at the root.
;
; For example: (consecutive-keys? (list 42 27) BST42) would produce true
; and (consecutive-keys? (list 10 3 1) BST10) would produce true
; but (consecutive-keys? (list 27 42) BST42) would produce false
;