```
1  (require spd/tags)
2
3  (@assignment bank/htdd-p9)
4  (@cwl ???)
5
6  ;; =================
7  ;; Data definitions:
8
9  (@problem 1)
10 ;; You are working on a system that will automate delivery for
11 ;; YesItCanFly! airlines catering service.
12 ;; There are three dinner options for each passenger, chicken, pasta
13 ;; or no dinner at all.
14 ;;
15 ;; Design a data definition to represent a dinner order. Call the type
16 ;; DinnerOrder.
17
18
19 (@htdd DinnerOrder)
20 ;; DinnerOrder is one of:
21 ;;  - "No dinner"
22 ;;  - "Chicken"
23 ;;  - "Pasta"
24 ;; interp. "No dinner" means the passenger does not want dinner,
25 ;;         the other values are dinner options
26 ;; <examples are redundant for enumerations>
27
28 (@dd-template-rules one-of          ;3 cases
29                     atomic-distinct ;"No dinner"
30                     atomic-distinct ;"Chicken"
31                     atomic-distinct) ;"Pasta"
32
33
34 (define (fn-for-dinner-order d)
35   (cond [(string=? d "No dinner") (...)]
36         [(string=? d "Chicken") (...)]
37         [(string=? d "Pasta") (...)]))
38
39
40
41 ;; =================
42 ;; Functions:
43
44 (@problem 2)
45 ;; Design the function dinner-order-to-msg that consumes a dinner order
46 ;; and produces a message for the flight attendants saying what the
47 ;; passenger ordered.
48 ;;
49 ;; For example, calling dinner-order-to-msg for a chicken dinner would
50 ;; produce "The passenger ordered chicken."
51
52
53 (@htdf dinner-order-to-msg)
54 (@signature DinnerOrder -> String)
55 ;; produce message to describe what passenger ordered
56 (check-expect (dinner-order-to-msg "No dinner")
57               "The passenger did not order dinner.")
58 (check-expect (dinner-order-to-msg "Chicken")
59               "The passenger ordered chicken.")
60 (check-expect (dinner-order-to-msg "Pasta")
61               "The passenger ordered pasta.")
62
63 ;(define (dinner-order-to-msg d) "d")  ;stub
```

A

B

C

D

E

F

G

```
64
65 (@template-origin DinnerOrder)
66
67 (@template
68  (define (dinner-order-to-msg d)
69    (cond [(string=? d "No dinner") (...)]
70          [(string=? d "Chicken") (...)]
71          [(string=? d "Pasta") (...)])))
72
73 (define (dinner-order-to-msg d)
74   (cond [(string=? d "No dinner") "The passenger did not order dinner."]
75         [(string=? d "Chicken") "The passenger ordered chicken."]
76         [(string=? d "Pasta") "The passenger ordered pasta."]))
```

H

I

J