```racket
;; CS 135 :: Fall 2017 :: Posted solution :: A02 :: cond.rkt

;; 1(a)

(define (q1a x)
  (cond
    [(and (p2? x) (p1? x)) 'left]
    [(p2? x) 'down]
    [(p1? x) 'up]
    [else 'right]))


;; 1(b)

(define (q1b x)
  (cond
    [(and (p1? x) (p2? x) (p1? (+ x 1))) 'up]
    [(and (p1? x) (p2? x) (p2? (* 2 x))) 'down]
    [(and (p1? x) (p2? x)) 'right]
    [(and (p1? x) (p2? 2)) 'down]
    [(p1? x) 'up]
    [(and (p1? 0) (p2? x)) 'left]
    [(p1? 0) 'right]
    [else 'down]))


;; 1(c)

(define (q1c x)
  (cond
    [(and (p1? x) (not (p2? x))) 'down]
    [else 'up]))



;; CS 135 :: Fall 2017 :: Posted solution :: A02 :: airmiles.rkt

(define standard-sponsor 15)
(define standard-nonsponsor 20)
(define premium-sponsor 10)
(define premium-nonsponsor 15)


;; (calc-airmiles amount card-type sponsor?) calculates the
;;   number of airmiles earned based on the total amount spent,
;;   the type of card indicated by card-type and whether
;;   or not the store was a sponsor
;; calc-airmiles: Num (anyof 'standard 'premium) Bool -> Nat
;; requires:  amount >= 0
;; Examples:
(check-expect (calc-airmiles 30 'standard false) 1)
(check-expect (calc-airmiles 22 'premium true) 2)

(define (calc-airmiles amount card-type sponsor?)
  (cond
    [sponsor? (cond
                [(symbol=? card-type 'standard)
                 (floor (/ amount standard-sponsor))]
```

```
                [else (floor (/ amount premium-sponsor))])]
    [(symbol=? card-type 'standard)
     (floor (/ amount standard-nonsponsor))]
    [else (floor (/ amount premium-nonsponsor))]))

;; Tests:
(check-expect (calc-airmiles 14 'standard true) 0)
(check-expect (calc-airmiles 22 'standard true) 1)
(check-expect (calc-airmiles 30 'premium false) 2)
(check-expect (calc-airmiles 117.25 'premium true) 11)
(check-expect (calc-airmiles 170.39 'premium false) 11)



;; CS 135 :: Fall 2017 :: Posted solution :: A02 :: grades.rkt

(define passing-grade 50)
(define part-weight 0.05)
(define assign-weight 0.2)
(define mid1-weight 0.1)
(define mid2-weight 0.2)
(define final-weight 0.45)
(define highest-failing 46)


;; (weighted-average mid1 mid2 fin-exam) computes the weighted average of the
;;   midterms, mid1 and mid2, and the final exam, fin-exam
;; weighted-average: Nat Nat Nat -> Num
;; requires: mid1 <= 100
;;           mid2 <= 100
;;           fin-exam <= 100
;; Examples:
(check-expect (weighted-average 0 100 100) (* 100 (/ 65 75)))
(check-expect (weighted-average 50 50 50) 50)

(define (weighted-average mid1 mid2 fin-exam)
  (/ (+ (* mid1 mid1-weight) (* mid2 mid2-weight) (* fin-exam final-weight))
     (+ mid1-weight mid2-weight final-weight)))


;; (normal-cs135-grade mid1 mid2 exam assign part) produces the numerical grade
;;   in cs135 based on the grades, mid1, mid2, exam, assign, and part,
;;   ignoring any failure conditions
;; normal-cs135-grade: Nat Nat Nat Nat Nat -> Num
;; requires:  mid1 <= 100
;;            mid2 <= 100
;;            part <= 100
;;            assign <= 100
;;            exam <= 100
;; Examples:
(check-expect (normal-cs135-grade 80 80 80 80 80) 80)
(check-expect (normal-cs135-grade 100 49 100 100 100) 89.8)

(define (normal-cs135-grade mid1 mid2 exam assign part)
  (+ (* mid1 mid1-weight) (* mid2 mid2-weight) (* exam final-weight)
     (* part part-weight) (* assign assign-weight)))
```

```
;; (final-cs135-grade mid1 mid2 exam assign part) produces the final grade in
;;   in cs135 based on the grades, mid1, mid2, exam, assign, and part,
;; final-cs135-grade: Nat Nat Nat Nat Nat -> Num
;; requires:  mid1 <= 100
;;            mid2 <= 100
;;            part <= 100
;;            assign <= 100
;;            exam <= 100
;; Examples:
(check-expect (final-cs135-grade 80 80 80 80 80) 80)
(check-expect (final-cs135-grade 49 49 49 100 100) highest-failing)

(define (final-cs135-grade mid1 mid2 exam assign part)
  (cond
    [(and (or (< assign passing-grade)
              (< (weighted-average mid1 mid2 exam) passing-grade))
          (< highest-failing (normal-cs135-grade mid1 mid2 exam assign part)))
     highest-failing]
    [else (normal-cs135-grade mid1 mid2 exam assign part)]))

;; Tests:
(check-expect (final-cs135-grade 50 50 44 50 50) highest-failing)
(check-expect (final-cs135-grade 50 50 40 50 50) 45.5)
(check-expect (final-cs135-grade 0 0 0 0 0) 0)
(check-expect (final-cs135-grade 0 0 0 0 100) 5)
(check-expect (final-cs135-grade 0 0 0 100 100) 25)
(check-expect (final-cs135-grade 0 0 100 0 100) highest-failing)
(check-expect (final-cs135-grade 0 0 100 0 0) 45)
(check-expect (final-cs135-grade 50 50 50 50 100) 52.5)
(check-expect (final-cs135-grade 49 49 49 100 100) highest-failing)
(check-expect (final-cs135-grade 100 100 0 100 100) highest-failing)
(check-expect (final-cs135-grade 100 100 100 100 100) 100)
(check-expect (final-cs135-grade 80.17 80.17 80.17 80.17 80.17) 80.17)
(check-expect (final-cs135-grade 45.9 45.9 45.9 45.9 45.9) 45.9)



;; CS 135 :: Fall 2017 :: Posted solution :: A02 :: blood.rkt

;; 4(a)

;; (can-donate-to/cond? donor recip) determines whether the blood type
;;   of donor is compatible with the blood type of recip
;;   can-donate-to/cond?:  Sym Sym -> Bool
;; requires:  donor is one of 'O- 'O+ 'A- 'A+ 'B- 'B+ 'AB+ 'AB-
;;            recip is one of 'O- 'O+ 'A- 'A+ 'B- 'B+ 'AB+ 'AB-
;; Examples:
(check-expect (can-donate-to/cond? 'O- 'O-) true)
(check-expect (can-donate-to/cond? 'O+ 'A-) false)

(define (can-donate-to/cond? donor recip)
  (cond [(symbol=? donor recip) true]
        [(symbol=? donor 'O-) true]
        [(symbol=? recip 'AB+) true]
        [(symbol=? donor 'O+) (cond [(symbol=? recip 'A+) true]
                                    [(symbol=? recip 'B+) true]
                                    [else false])]
        [(symbol=? donor 'A-) (cond [(symbol=? recip 'A+) true]
```

```
                                    [(symbol=? recip 'AB-) true]
                                    [else false])]
        [(symbol=? donor 'B-) (cond [(symbol=? recip 'B+) true]
                                    [(symbol=? recip 'AB-) true]
                                    [else false])]
        [else false]))

;; Tests:
(check-expect (can-donate-to/cond? 'O- 'A+) true)
(check-expect (can-donate-to/cond? 'O+ 'AB+) true)
(check-expect (can-donate-to/cond? 'O+ 'A+) true)
(check-expect (can-donate-to/cond? 'O+ 'B+) true)
(check-expect (can-donate-to/cond? 'A- 'A+) true)
(check-expect (can-donate-to/cond? 'A- 'AB-) true)
(check-expect (can-donate-to/cond? 'A- 'O+) false)
(check-expect (can-donate-to/cond? 'B- 'B+) true)
(check-expect (can-donate-to/cond? 'B- 'AB-) true)
(check-expect (can-donate-to/cond? 'B- 'O+) false)
(check-expect (can-donate-to/cond? 'B+ 'O+) false)


;; 4(b)

;; (can-donate-to/bool? donor recip) determines whether the blood type
;;   of donor is compatible with the blood type of recip
;; can-donate-to/bool?:  Sym Sym -> Bool
;; requires:  donor is one of 'O- 'O+ 'A- 'A+ 'B- 'B+ 'AB+ 'AB-
;;            recip is one of 'O- 'O+ 'A- 'A+ 'B- 'B+ 'AB+ 'AB-
;; Examples:
(check-expect (can-donate-to/bool? 'O- 'O-) true)
(check-expect (can-donate-to/bool? 'O+ 'A-) false)

(define (can-donate-to/bool? donor recip)
  (or (symbol=? donor recip)
      (symbol=? donor 'O-)
      (symbol=? recip 'AB+)
      (and (symbol=? donor 'O+)
           (or (symbol=? recip 'A+)
               (symbol=? recip 'B+)))
      (and (symbol=? donor 'A-)
           (or (symbol=? recip 'A+)
               (symbol=? recip 'AB-)))
      (and (symbol=? donor 'B-)
           (or (symbol=? recip 'B+)
               (symbol=? recip 'AB-)))))

;; Tests:
(check-expect (can-donate-to/bool? 'O- 'A+) true)
(check-expect (can-donate-to/bool? 'O+ 'AB+) true)
(check-expect (can-donate-to/bool? 'O+ 'A+) true)
(check-expect (can-donate-to/bool? 'O+ 'B+) true)
(check-expect (can-donate-to/bool? 'A- 'A+) true)
(check-expect (can-donate-to/bool? 'A- 'AB-) true)
(check-expect (can-donate-to/bool? 'A- 'O+) false)
(check-expect (can-donate-to/bool? 'B- 'B+) true)
(check-expect (can-donate-to/bool? 'B- 'AB-) true)
(check-expect (can-donate-to/bool? 'B- 'O+) false)
(check-expect (can-donate-to/bool? 'B+ 'O+) false)
```

```
(check-expect (can-donate-to/bool? 'B- 'AB-) true)
(check-expect (can-donate-to/bool? 'B- 'O+) false)
(check-expect (can-donate-to/bool? 'B+ 'O+) false)
```