```
;; CS 135 :: Fall 2017 :: Posted solution :: A01 :: functions.rkt

;; 1(a)

(define (distance x1 y1 x2 y2)
  (+ (abs (- x1 x2))
     (abs (- y1 y2))))


;; 1(b)

(define (Stirling n)
  (* (expt n
           (+ n 1/2))
     (exp (- 1 n))))


;; 1(c)

(define (logit p)
  (log (/ p
          (- 1 p))))


;; 1(d)

(define (freq base-frequency interval)
  (* base-frequency
     (expt 2 (/ interval 12))))


;; 1(e)

(define (d1 maturity rate volatility spot-price strike-price)
  (* (/ 1 (* volatility (sqrt maturity)))
     (+ (log (/ spot-price strike-price))
        (* (+ rate
              (/ (sqr volatility) 2))
           maturity))))


;; 1(f)

(define g 9.8)

(define (height initial-velocity time)
  (- (* initial-velocity time)
     (* (* 1/2 g) (sqr time))))



;; CS 135 :: Fall 2017 :: Posted solution :: A01 :: conversion.rkt

;; 2(a)

(define metres/mile 1609.344)
(define seconds/hour 3600)
```

```
;; (m/s->mph speed-m/s) converts the given speed-m/s in units of
;;   m/s to the equivalent speed in units of mph
;; m/s->mph: Num -> Num
;; Example:
(check-expect (m/s->mph 1) 3600000/1609344)

(define (m/s->mph speed-m/s)
  (/ (* speed-m/s seconds/hour)
     metres/mile))

;; Tests:
(check-expect (m/s->mph 0) 0)
(check-expect (m/s->mph 1609.344) 3600)


;; 2(b)

(define metres/Smoot 1.7018)
(define seconds/millifortnight 1209.6)
(define Smoots/mile (/ metres/mile metres/Smoot))
(define hours/millifortnight (/ seconds/millifortnight seconds/hour))


;; (mph->S/mfn speed-mph) converts the given speed-mph in units of mph
;;     to the equivalent speed in units of Smoots per millifortnight
;; mph->S/mfn: Num -> Num
;; Example:
(check-expect (mph->S/mfn 3600000/1609344) 12096000/17018)

(define (mph->S/mfn speed-mph)
  (* (* speed-mph Smoots/mile) hours/millifortnight))

;; Tests:
(check-expect (mph->S/mfn 0) 0)
(check-expect (mph->S/mfn 17018/12096000) 1609344/3600000)
(check-expect (mph->S/mfn 1) 19466625024/61264800)


;; 2(c)

(define metres/100km 100000)
(define L/gal 3.785411784)
(define miles/100km (/ metres/100km metres/mile))


;; (mpg->L/100km efficiency-mpg) converts a fuel efficiency-mpg
;;     in units of mpg to the equivalent efficiency in units of L/100km
;; mpg->L/100km: Num -> Num
;; requires: efficiency-mpg > 0
;; Example:
(check-expect (mpg->L/100km 1) 3785411784/16093440)

(define (mpg->L/100km efficiency-mpg)
  (* (/ L/gal efficiency-mpg) miles/100km))

;; Tests:
(check-expect (mpg->L/100km 10) 3785411784/160934400)
```

```
(check-expect (mpg->L/100km 1/48) 11290.3)
(check-expect (mpg->L/100km 635/24) 8.89)


;; CS 135 :: Fall 2017 :: Posted solution :: A01 :: grades.rkt


(define midterm1-weight     10/100)
(define midterm2-weight     20/100)
(define final-weight        45/100)
(define assignment-weight   20/100)
(define participation-weight 5/100)
(define participation-grade  100)


;; 3(a)

;; (final-cs135-grade midterm1-grade midterm2-grade
;;                    final-grade assignment-grade)
;;   determines the final grade in CS135 based on the four components
;;   [midterm1-grade midterm2-grade final-grade assignment-grade]
;;   and perfect participation
;; final-cs135-grade: Nat Nat Nat Nat -> Num
;; requires: midterm1-grade   <= 100
;;           midterm2-grade   <= 100
;;           final-grade      <= 100
;;           assignment-grade <= 100
;; Examples:
(check-expect (final-cs135-grade 100 100 100 100) 100)
(check-expect (final-cs135-grade 0 0 0 0) 5)

(define (final-cs135-grade midterm1-grade midterm2-grade
                           final-grade assignment-grade)
  (+ (* midterm1-grade midterm1-weight)
     (* midterm2-grade midterm2-weight)
     (* final-grade final-weight)
     (* assignment-grade assignment-weight)
     (* participation-grade participation-weight)))

;; Tests:
(check-expect (final-cs135-grade 100 0 0 0) 15)
(check-expect (final-cs135-grade 0 100 0 0) 25)
(check-expect (final-cs135-grade 0 0 100 0) 50)
(check-expect (final-cs135-grade 0 0 0 100) 25)
(check-expect (final-cs135-grade 50 50 50 50) 52.5)
(check-expect (final-cs135-grade 70 90 80 75) 81)


;; 3(b)

(define target-grade 60)


;; (cs135-final-exam-grade-needed midterm1-grade midterm2-grade
;;                                assignment-grade)
;;   determines the final exam grade needed to achieve a target-grade
;;   in CS135 based on [midterm1-grade midterm2-grade assignment-grade]
```

```
;;   and perfect participation
;; cs135-final-exam-grade-needed: Nat Nat Nat -> Num
;; requires: midterm1-grade   <= 100
;;           midterm2-grade   <= 100
;;           assignment-grade <= 100
;; Example:
(check-expect (cs135-final-exam-grade-needed 100 100 100) 100/9)

(define (cs135-final-exam-grade-needed midterm1-grade midterm2-grade
                                       assignment-grade)
  (/ (- target-grade
        (+ (* midterm1-grade midterm1-weight)
           (* midterm2-grade midterm2-weight)
           (* assignment-grade assignment-weight)
           (* participation-grade participation-weight)))
     final-weight))

;; Tests:
(check-expect (cs135-final-exam-grade-needed 0 0 0) 5500/45)
(check-expect (cs135-final-exam-grade-needed 100 0 0) 100)
(check-expect (cs135-final-exam-grade-needed 0 100 0) 700/9)
(check-expect (cs135-final-exam-grade-needed 0 0 100) 700/9)
(check-expect (cs135-final-exam-grade-needed 70 90 75) 100/3)
```