```
;;
;; ************************************************
;; Sample Solutions
;; CS 135 Fall 2019
;; Assignment 02, Problem 1
;; ************************************************
;;


;; 1a)

(define (q1a-alt n a?)
  (cond
    [(and a? (>= n 0)) (add1 n)]
    [(and a? (< n 0)) (sub1 n)]
    [else 0]))


;; 1b)

(define (q1b-alt a? b? c?)
  (cond
    [(and a? b?) 'elm]
    [(and a? (not c?)) 'birch]
    [a? 'cedar]
    [b? 'pine]
    [(not c?) 'birch]
    [else 'cherry]))


;; 1c)

(define (q1c-alt a? b? c?)
  (cond
    [(and c? a?) 'oak]
    [(and c? b?) 'maple]
    [c? 'willow]
    [(and a? b?) 'walnut]
    [a? 'dogwood]
    [(not b?) 'sumac]
    [else 'buckthorn]))


;; 1d)

(define (q1d-alt a? b? c?)
  (cond
    [(and (not a?) (not b?) (not c?)) 'larch]
    [(and (not a?) (not b?)) 'hickory]
    [(and a? (not (and b? c?))) 'hazel]
    [else 'spruce]))



;;
;; ************************************************
;; Sample Solutions
;; CS 135 Fall 2019
;; Assignment 02, Problem 2
;; ************************************************
;;
```

```
(define name-length-req 10)
(define name-length-mod 25)
(define name-length-default 0)
(define name-start-req "The")
(define name-start-mod -50)
(define name-start-default 0)
(define studio-mod-good 500)
(define studio-good "Marvel")
(define studio-mod-bad -250)
(define studio-bad "DC")
(define studio-default 0)
(define actor-mod 50)
(define explosion-mod 6)
(define explosion-base -20)


;;(name-mod name) produces the affect the name of a movie has on its profits
;; name-mod: Str -> Num
;; Example:
(check-expect (name-mod "Short") 25)

(define (name-mod name)
  (+ (cond
       [(< (string-length name) name-length-req) name-length-mod]
       [else name-length-default])
     (cond
       [(string=? (substring name 0 (min (string-length name)
                                         (string-length name-start-req)))
                  name-start-req) name-start-mod]
       [else name-start-default])))


;;(studio-mod studio) produces the affect the studio has on a movie's profits
;; studio-mod: Str -> Num
;; Example:
(check-expect (studio-mod "DC") -250)

(define (studio-mod studio)
  (cond
       [(string=? studio studio-good) studio-mod-good]
       [(string=? studio studio-bad) studio-mod-bad]
       [else studio-default]))


;;(box-office-profits name studio actors explosions) produces the projected
;;   box office profit of a movie given its name, studio, number of famous
;;   actors, and number of explosions
;; box-office-profits: Str Str Nat Nat -> Num
;; Examples
(check-expect (box-office-profits "Avengers: more endgames" "Marvel" 4 50 ) 980)
(check-expect (box-office-profits "Superman v Superman" "DC" 2 100 ) 430)
(check-expect (box-office-profits "The Slog" "New Line Cinema" 0 0 ) -45)

(define (box-office-profits name studio actors explosions)
  (+ (name-mod name)
     (studio-mod studio)
     (* actors actor-mod)
     (+ (* explosions explosion-mod) explosion-base)))

;; Tests
(check-expect (box-office-profits "Avengers: more endgames" "Marvel" 4 50 ) 980)
(check-expect (box-office-profits "Superman" "DC" 2 100 ) 455)
(check-expect (box-office-profits "The Slog" "New Line Cinema" 0 0 ) -45)
```

```
(check-expect (box-office-profits "BO" "New Line Cinema" 0 0 ) 5)
(check-expect (box-office-profits "ten letter" "New Line Cinema" 0 0) -20)


;;
;; ***************************************************
;; Sample Solutions
;; CS 135 Fall 2019
;; Assignment 02, Problem 3
;; ***************************************************
;;


;; 3a)

;; (intentional-grounding? pressure pocket team-member) determines if a
;;   quarterback throwing the ball counts as intentional grounding while
;;   they are under pressure and in the pocket, and if there was an
;;   available team-member
;; intentional-grounding?: Bool Bool Bool -> Bool
;;Examples:
(check-expect (intentional-grounding? false true false) false)
(check-expect (intentional-grounding? true true false) true)

(define (intentional-grounding? pressure pocket team-member)
  (and pressure pocket (not team-member)))

;; Tests:
(check-expect (intentional-grounding? true false true) false)
(check-expect (intentional-grounding? true true true) false)


;; 3b)

;; (intentional-grounding-correct? pressure pocket team-member eligible)
;;   determines if a quarterback throwing the ball counts as intentional
;;   grounding while they are under pressure and in the pocket, and if there
;;   was an available team member who was eligible
;; intentional-grounding-correct?: Bool Bool Bool Bool -> Bool
;;Examples:
(check-expect (intentional-grounding-correct? false true false false) false)
(check-expect (intentional-grounding-correct? true true true false) true)

(define (intentional-grounding-correct? pressure pocket team-member eligible)
  (intentional-grounding? pressure pocket (and team-member eligible)))

;; Tests:
(check-expect (intentional-grounding-correct? false false false false) false)
(check-expect (intentional-grounding-correct? true false false false) false)
(check-expect (intentional-grounding-correct? true true false true) true)
(check-expect (intentional-grounding-correct? true true true true) false)
(check-expect (intentional-grounding-correct? true true true false) true)


;; 3c)

;; (intentional-grounding-penalty pressure pocket team-member eligible endzone)
;;   determines the intentional grounding penalty for a quarterback throwing the ball
;;   while they are under pressure and in the pocket and in the endzone, when there
;;   was an available team-member who was eligible
;; intentional-grounding-penalty: Bool Bool Bool Bool Bool -> Sym
;;Examples:
(check-expect (intentional-grounding-penalty true true true false false) '10yds)
```

```
(check-expect (intentional-grounding-penalty true true false false true) 'Safety)

(define (intentional-grounding-penalty pressure pocket team-member eligible endzone)
  (cond
    [(and (intentional-grounding-correct? pressure pocket team-member eligible) endzone)
'Safety]
    [(intentional-grounding-correct? pressure pocket team-member eligible) '10yds]
    [else 'None]))

;; Tests:
(check-expect (intentional-grounding-penalty false false true false false) 'None)
(check-expect (intentional-grounding-penalty true false true false false) 'None)
```