1. $|\emptyset| = 0$, $|[\emptyset]| = 1$ and $|[\mathbb{N}]| = 1$.

2. a  count $\leftarrow 0$
   while $(\text{count} \geq 0) \land (\text{input not empty})$
   if next input is $S$ then count $\text{++}$ else count $\text{--}$
   if $(\text{count} = 0) \land (\text{input empty})$ then input admissible else input not admissible

   b  Algorithm $SSSSXSSXSSSSXSSXX$ produces $(3, 5, 7, 6, 8, 4, 9, 2, 10, 1)$, but any
       algorithm to produce $(3, 1, 2)$ must push 1 and 2 and then do $SX$ upon seeing 3. Hence it
       must start $SSXX$. But 2 was pushed onto the stack after 3 (hence it’s on top) and there’s no
       way to print 1 before 2.

   c  The elements in the stack are sorted in decreasing order, from largest at the top to
       smallest at the bottom.

Extra Credit: $(a_1, \ldots, a_n)$ is realizable if and only if there do not exist $1 \leq i < j < k \leq n$
       such that $a_j < a_k < a_i$.

3. A combination to make change is a vector $(v, w, x, y, z)$ where $v$ (respectively $w, x, y$ and $z$)
       is the number of 15¢ (respectively 23¢, 29¢, 41¢ and 67¢) coins. We can make change
       for $n$¢ in any of the following ways:
       • $(v+1, w, x, y, z)$ if $(v, w, x, y, z)$ makes change for $(n-15)$¢
       • $(v, w+1, x, y, z)$ if $(v, w, x, y, z)$ makes change for $(n-23)$¢
       • $(v, w, x+1, y, z)$ if $(v, w, x, y, z)$ makes change for $(n-29)$¢
       • $(v, w, x, y+1, z)$ if $(v, w, x, y, z)$ makes change for $(n-41)$¢
       • $(v, w, x, y, z+1)$ if $(v, w, x, y, z)$ makes change for $(n-67)$¢

As initialization, there is no way to make change for $n$¢ if $n < 0$.

(define-struct tally (v w x y z))

;; these 5 functions update the counts in the tally (without using assignment)
(define (incr-v a-tally) (make-tally (+ 1 (tally-v a-tally))
  (tally-w a-tally)
  (tally-x a-tally)
  (tally-y a-tally)
  (tally-z a-tally)))

(define (incr-w a-tally) (make-tally (tally-v a-tally)
  (+ 1 (tally-w a-tally))
  (tally-x a-tally)
  (tally-y a-tally)
  (tally-z a-tally)))


(define (incr-x a-tally)
  (make-tally (tally-v a-tally)
    (tally-w a-tally)
    (+ 1 (tally-x a-tally))
    (tally-y a-tally)
    (tally-z a-tally)))

(define (incr-y a-tally)
  (make-tally (tally-v a-tally)
    (tally-w a-tally)
    (tally-x a-tally)
    (+ 1 (tally-y a-tally))
    (tally-z a-tally)))

(define (incr-z a-tally)
  (make-tally (tally-v a-tally)
    (tally-w a-tally)
    (tally-x a-tally)
    (tally-y a-tally)
    (+ 1 (tally-z a-tally))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; the brute-force, one solution version

;; change : number -> tally or false
;; returns spread of coins totaling amount, or false if no combo of coins
;; adds up to the given amount
(define (change amount)
  (cond [(< amount 0) false]
        [(= amount 0) (make-tally 0 0 0 0 0)]
        [else (let ([coins (change (- amount 15))])
            (cond [((tally? coins) (incr-v coins))
                  (else (let ([coins (change (- amount 23))])
                          (cond [((tally? coins) (incr-w coins))
                                  (else (let ([coins (change (- amount 29))])
                                          (cond [((tally? coins) (incr-x coins))
                                                  (else (let ([coins (change (- amount 41))])
                                                          (cond [((tally? coins) (incr-y coins))
                                                                  (else (let ([coins (change (- amount 67))])
                                                                          (cond [((tally? coins) (incr-z coins))
                                                                                  [else false]])))))))))))))]))])])})