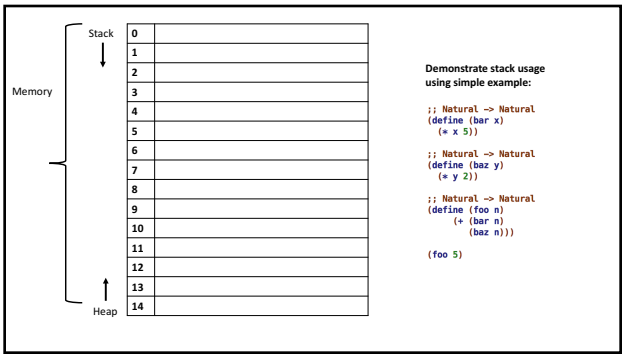


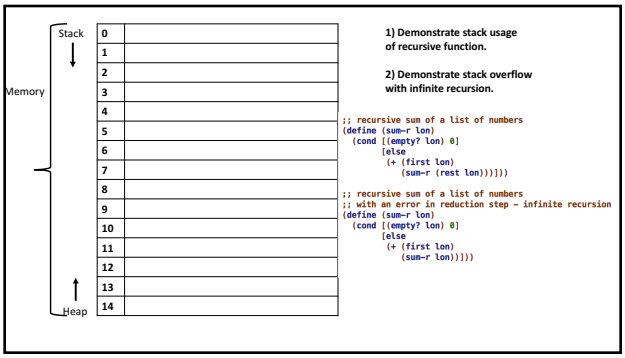
Demonstrate stack usage using simple example:

```
;; Natural -> Natural
(define (bar x)
  (* x 5))

;; Natural -> Natural
(define (baz y)
  (* y 2))

;; Natural -> Natural
(define (foo n)
     (+ (bar n)
        (baz n)))

(foo 5)
```

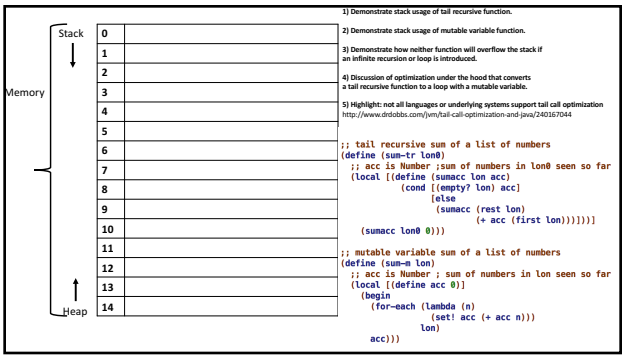


1) Demonstrate stack usage of recursive function.

2) Demonstrate stack overflow with infinite recursion.

```
;; recursive sum of a list of numbers
(define (sum-r lon)
  (cond [(empty? lon) 0]
        [else
         (+ (first lon)
            (sum-r (rest lon))))))

;; recursive sum of a list of numbers
;; with an error in reduction step - infinite recursion
(define (sum-r lon)
  (cond [(empty? lon) 0]
        [else
         (+ (first lon)
            (sum-r lon))]))
```



1) Demonstrate stack usage of tail recursive function.

2) Demonstrate stack usage of mutable variable function.

3) Demonstrate how neither function will overflow the stack if an infinite recursion or loop is introduced.

4) Discussion of optimization under the hood that converts a tail recursive function to a loop with a mutable variable.

5) Highlight: not all languages or underlying systems support tail call optimization
http://www.drdobbs.com/jvm/tail-call-optimization-and-java/240167044

```
;; tail recursive sum of a list of numbers
(define (sum-tr lon0)
  ;; acc is Number ;sum of numbers in lon0 seen so far
  (local [(define (sumacc lon acc)
            (cond [(empty? lon) acc]
                  [else
                   (sumacc (rest lon)
                           (+ acc (first lon)))))))]
    (sumacc lon0 0)))

;; mutable variable sum of a list of numbers
(define (sum-m lon)
  ;; acc is Number ; sum of numbers in lon seen so far
  (local [(define acc 0)]
    (begin
      (for-each (lambda (n)
                  (set! acc (+ acc n)))
                lon)
      acc)))
```