Q1 – (a)

```
(define Find_Help
   (lambda (x lst)
      (if (null? lst) (list)
            (if (= x (car lst)) (car lst)
                  (if (null? (cdr lst))
                        (list)
                        (Find_Help x (cdr lst)))))))


(define Find
   (lambda (x lst)
      (if (null? lst)
            (list)
            (if (null? (Find_Help x (car lst)))
                  (Find x (cdr lst))
                  (if (= x (Find_Help x (car lst)))
                        (car lst)
                        (if (null? (cdr lst))
                              (list)
                              (Find x (cdr lst))))))))
```

Q1 – (b)

```
(define (append-test lhs rhs)
   (if (empty? lhs)
         rhs
         (cons (first lhs) (append-test (rest lhs) rhs))))


(define Concatenate
   (lambda (lst)
      (if (null? lst)
            lst
            (append-test (car lst) (Concatenate (cdr lst))))))
```

Q2

```
(define (deleteItem lst item)
   (cond ((null? lst)
              '())
             ((equal? item (car lst))
              (cdr lst))
             (else
              (cons (car lst)
                       (deleteItem (cdr lst) item)))))


(define getElement
   (lambda (n lst)
      (if (= n 0) (car lst)
             (getElement (- n 1) (cdr lst)))))
(define Random
   (lambda (lst)
      (let ((x (length lst)))
       (let ((r (random x)))
          (getElement r lst)))))
(define shuffle
   (lambda (lst)
      (if (null? lst)
             lst
             (shuffle_help lst (Random lst)))))


(define shuffle_help
```

```
(lambda (lst ele)
   (if (null? (cdr lst))
         lst
         (cons ele (shuffle (deleteItem lst ele))))))
```