Problem #1 (5 points):
What will Scheme print in response to the following expressions? If an expression produces an error message, you may just say error; you don't have to provide the exact text of the message. If the value of an expression is a procedure, just say procedure; you don't have to show the form in which Scheme prints procedures.

(let ((a 3) (b 4))
(lambda () (+ a b)))

(let ((a 3) (b 4))
((lambda () (* a b))))

(every - (filter number? '(the 1 after 909))) ; EVERY from homework 2

For the following, also draw a box and pointer diagram of the value produced by each expression.

(cons (a b) (list (c d) e))

(cddar ((a b c) (d e f) (g h i)))

Problem #2 (2 points)
(a) Indicate the order of growth in time of foo below:

(define (foo n)
  (if (< n 2)
    1
    (+ (baz (- n 1))
        (baz (- n 2)))))

(define (baz n)
  (+ n (- n 1)))

___Theta(1)  ___Theta(n)  ___Theta(n^2)  ___Theta(2^n)
(b) Indicate the order of growth in time of garply below:

\[
\text{(define (garply n)}
\quad \text{(if (= n 0)}
\quad \quad 0
\quad \text{(+ (factorial n) (garply (- n 1)))))}
\]

\[
\text{(define (factorial n)}
\quad \text{(if (= n 0)}
\quad \quad 1
\quad \quad (\ast n \text{(factorial (- n 1))}})
\]

___Theta(1)    ___Theta(n)    ___Theta(n^2)    ___Theta(2^n)

Problem #3 (2 points)
If an expression produces an error, just say "error": if it returns a procedure, just say "procedure."

Given the following definitions:

\[
\text{(define (mountain x) 'done)}
\]
\[
\text{(define (dew) (dew))}
\]

(a) What will be the result of the expression (mountain (dew))
   in normal order? _____________
   in applicative order? ____________

(b) What will be the result of the expression (mountain dew)
   in normal order? _____________
   in applicative order? ____________

Problem #4 (2 points)

\[
\text{(define (even? n)}
\quad \text{(cond ((= n 0) #t)}
\quad \quad ((= n 1) #f)
\quad \quad (else (if (even ? (- n 2))
\quad \quad \quad #t
\quad \quad \quad #f)))}
\]

Does this procedure generate an iterative process or a recursive process?

If iterative, explain why in one sentence. If recursive, rewrite it, changing as little as possible, to make it
generate an iterative process.
Problem #5 (4 points)
This question concerns the twenty-one game used in the first programming project.
(Assume the version without jokers.)

(a) Write a procedure random-strategy that takes a list of strategies as its argument, and returns a strategy that randomly uses one of the strategies from the list each time it's called. You may use this helper procedure:

(define (pick seq)
  (list-ref seq (random (length seq)))))

(b) Using the procedures every (from homework 2) and/or filter (from lecture), write a strategy called lovelorn that asks for an additional card if and only if there are no hearts in the hand.

Problem #6 (4 points)
The following partly-written procedure takes a list of sentences as its argument. It should return a sentence containing the first word of the first sentence, the second word of the second sentence, and so on. (Assume the sentences are long enough; don't add error checks.)

> (diagonal '((she loves you) (tell me why) (i want to hold your hand)))
(she me to)

Fill in the blanks to complete the definitions correctly. Respect the data abstraction: use sentence procedures for sentences, list procedures for lists.

(define (diagonal lstsents)
  (if (_____ lstsents)
      '()
      (_________ (_________ (_________ lstsents))
      (diagonal (chop (_____ lstsents))))))

(define (chop lstsents) ; Remove first word from each sentence
  (if (_____ lstsents)
      '()
      (_______ (_______ (_______ lstsents))
      (chop (_______ lstsents))))