Question #1
What will Scheme print in response to the following expressions? If it produces an error or runs forever without a result, just say "error". If it is a procedure, say "procedure". Assume no global variables have been defined beforehand except where noted.

(word '(+2 3) (+2 3))

((lambda (x y z) (* 5 y)) 3 4 7)

:from ex. 1.32, p. 61
(accumulate se '(hurrah) (lambda(x) (word 'hip x)) 1 (lambda (x) (1+x)) 3)

((if 3 - *) 32 2)

(a b c)

(let ((a 5) (* +) (+ *))
  (+ a a))

((lambda (-) (- 2)) (lambda (*)(+ * 4)))

Question #2 (True or False?)
_______ A $\mathcal{O}(\log(2n))$ algorithm is slower than a $\mathcal{O}(2 \log(n))$ algorithm
_______ For small size inputs knowing the $\mathcal{O}$ order of an algorithm is more useful than for large inputs
_______ If $f(x)$ is $\mathcal{O}(\log x)$, then $\lim (x \rightarrow \infty) f(x)/(\log x)$ is zero
_______ If $f$ is defined as $(define (f x) (* x x x))$ then and applicative-order evaluation of $(f (g y))$ evaluates $(g y)$ more often than a normal -order evaluation
_______ Function $g$ below defines a linear recursive process

  (define (g a b c) (if (> a b) c (+ c ( g (+ a 1) (- b 1) (+ c 1)))))

Question #3
Write a linear iterative function $li\text{-}nth$ that takes a number $n$ and a sentence and returns the $n$'th element of that sentence and an empty sentence if there is no such element. Count from zero.

For example,

> (li\text{-}nth 0 '(1 2 3 4))
1
> (li\text{-}nth 2 '(1 2 3 4))
3
> (li\text{-}nth 4 '(1 2 3 4))
'()
Question #4

`random` takes an argument n, and returns a random number from the set {0, 1, ..., n-1}.

Does the following segment of scheme define a function? In one sentence, explain why or why not.

```
(define (f x)
  ((lambda (u)
    (let ((a 0)
          (b 1))
      (random u)))
  (+ 45 x))
```

Question #5

A polynomial can be repeated as a sentence, where the words are the coefficients of the terms.
The first element of the sentence represents the term of degree 0 (the constant term), the second
represents the term of degree 1, etc. So, for example, $3x^2 + 2x + 1$ would be '(1 2 3) and $27x^8 + 1$
would be '(1 0 0 0 0 0 0 0 27) The polynomial whose coefficients are all zero is represented by '().

Write a function `add-polys` that takes two polynomials each of arbitrary degree, each represented
as a sentence and returns their sum, represented as a sentence. For example

```
> (add-polys '(1 2 3) '(1 0 0 0 0 0 0 0 27))
(2 2 3 0 0 0 0 0 27)
```

In this representation, multiplying polynomials by terms (otherwise known as monomials; $9x^2$ is a
monomial or term whereas $9x^2 + 1$ is a polynomial with two terms) involves shifting and multiplying.
Write a function `term-multiply-poly` that takes a polynomial of arbitrary degree represented as a
sentence, a term coefficient and the degree of a term and returns the product represented by a sentence
For example, if I wanted to multiply $9x^2 + 2x + 1$ by $7x^3$, I would do:

```
> (term-multiply-poly '(1 2 9) 7 3)
(0 0 0 7 14 63)
```

Question #6

Write a procedure `interleave-2` that takes two sentences s1 and s2 as its arguments. It returns the
sentence whose elements are alternate elements of s1 and s2 beginning with the first (the first of s2,
the first of s1, the second of s2, the second of s1, and so on). If one sentence is longer than the other,
it behaves as if the shorter were padded with 0's.

For example,

```
> (interleave-2 '(1 2 3 4) '(5 6 7 8))
(5 1 6 2 7 3 8 4)
> (interleave-2 '(9) '(10))
(10 9)
> (interleave-2 '(9) '(10 11 12))
(10 9 11 0 12 0)
> (interleave-2 '(1 2 3 4) '(5))
(5 1 0 2 0 3 0 4)
```
Question #7
Write a function `decapitate-and-keep-head` that, given a function $f$ of one argument returns a new function of one argument that returns the same value as $f(x)$, except the value is the first of what $f(x)$ would return
For example,

```
(define foo (decapitate-and-keep-head square))
```

> (foo 8)
6
> (foo 12)
1