Below is the Pig Latin code provided in lab.

```scheme
(define (pigl wd)
  (if (pl-done? wd)
      (word wd 'ay)
      (pigl (word (bf wd) (first wd))))
)

(define (pl-done? wd)
  (vowel? (first wd)))

(define (vowel? letter)
  (member? letter '(a e i o u)))
```

Q1: We LOVE helper procedures and think that you should too! But to test your understanding of how these helper procedures are working, please re-write the bolded code in pigl without calling the helper procedures pl-done? and vowel? Without changing the behavior of the function pigl, \((\text{pl-done? } \text{wd})\) can be replaced with:

Q2: Fill in the blank to show what scheme would print.
STk> (define (a b c)
    (if (= b 1)
        c
        (+ c (a (- b 1) c))))
a
STk> (a 4 7)

Q3: Write the procedure multiply that multiplies all of the numbers in a sentence as shown by the example calls below.
STk> (multiply '(1 2))
2
STk> (multiply '(10 3 2))
60
STk> (multiply '())
1

Q4: How many times is * called in the following code:

STk> (define (square x) (* x x))
STk> (define (weird x y) (* y y y y))
STk> (weird (square (* 1 1)) (* 3 3))

Using applicative order: ___________
Using normal order: ___________
Below is the Pig Latin code provided in lab.

```scheme
(define (pigl wd)
  (if (pl-done? wd)
      (word wd 'ay)
      (pigl (word (bf wd) (first wd)))))

(define (pl-done? wd)
  (vowel? (first wd)))

(define (vowel? letter)
  (member? letter '(a e i o u)))
```

Q1: We LOVE helper procedures and think that you should too! But to test your understanding of how these helper procedures are working, please re-write the bolded code in pigl without calling the helper procedures pl-done? and vowel? Without changing the behavior of the function pigl, (pl-done? wd) can be replaced with:

__________________________________________________________________________________________

Q2: Fill in the blank to show what scheme would print.

```scheme
STk> (define (a b c)
    (if (= b 1)
        c
        (+ c (a (- b 1) c))))
a
STk> (a 4 3)
```

Q3: Write the procedure multiply that multiplies all of the numbers in a sentence as shown by the example calls below.

```scheme
STk> (multiply '(1 2))
2
STk> (multiply '(10 3 2))
60
STk> (multiply '())
1
```

Q4: How many times is * called in the following code:

```scheme
STk> (define (square x) (* x x))  
STk> (define (weird x y) (* y y y y y))
STk> (weird (square (* 1 1)) (* 3 3))
```

Using applicative order: ____________

Using normal order: ____________
Below is the Pig Latin code provided in lab.

```scheme
(define (pigl wd)
  (if (pl-done? wd)
      (word wd 'ay)
      (pigl (word (bf wd) (first wd))))))

(define (pl-done? wd)
  (vowel? (first wd)))

(define (vowel? letter)
  (member? letter '(a e i o u)))

Q1: We LOVE helper procedures and think that you should too! But to test your understanding of how these helper procedures are working, please re-write the bolded code in pigl without calling the helper procedures pl-done? and vowel? Without changing the behavior of the function pigl, (pl-done? wd) can be replaced with:

Q2: Fill in the blank to show what scheme would print.
STk> (define (a b c)
    (if (= b 1)
      c
      (+ c (a (- b 1) c))))

a
STk> (a 4 6)

Q3: Write the procedure multiply that multiplies all of the numbers in a sentence as shown by the example calls below.
STk> (multiply '(1 2))
2
STk> (multiply '(10 3 2))
60
STk> (multiply '())
1

Q4: How many times is * called in the following code:

STk> (define (square x) (* x x))
STk> (define (weird x y) (* y y y y))
STk> (weird (square (* 1 1)) (* 3 3))

Using applicative order: ___________
Using normal order: ___________
Below is the Pig Latin code provided in lab.

```scheme
(define (pigl wd)
  (if (pl-done? wd)
      (word wd 'ay)
      (pigl (word (bf wd) (first wd)))))

(define (pl-done? wd)
  (vowel? (first wd)))

(define (vowel? letter)
  (member? letter '(a e i o u)))
```

Q1: We LOVE helper procedures and think that you should too! But to test your understanding of how these helper procedures are working, please re-write the bolded code in pigl without calling the helper procedures pl-done? and vowel? Without changing the behavior of the function pigl, \textbf{(pl-done? \textit{wd})} can be replaced with:

Q2: Fill in the blank to show what scheme would print.

```
STk> (define (a b c)
    (if (= b 1)
        c
        (+ c (a ( - b 1) c)))

a

STk> (a 4 5)
```

Q3: Write the procedure \textit{multiply} that multiplies all of the numbers in a sentence as shown by the example calls below.

```
STk> (multiply '(1 2))
2
STk> (multiply '(10 3 2))
60
STk> (multiply '())
1
```

Q4: How many times is * called in the following code:

```
STk> (define (square x) (* x x))
STk> (define (weird x y) (* y y y))
STk> (weird (square (* 1 1)) (* 3 3))
```

Using applicative order: ____________

Using normal order: ____________