Practice Quiz (02/14/2019)
Topics: Regular Expressions, List operations

1. What are the results to the following expressions?

```
> (define example "my name is enumerable")
> (define lst (list "5" "2" "6" "0" "1"))
> (regexp-match* #px"n[^a]*m." example)
```

```
> (string-split example #px"n.m")
```

```
> (regexp-replace* #px"([a-z]*) ([a-z]*)" example "\\1,\\2,\\1")
```

```
> (map (section + <> (reduce + (list 1 2 3))) lst)
```

```
> (map (o add1 string->number) lst)
```

2. Write regular expressions for each of the following.

    a. Words that contain two vowels in sequence.

    b. Two words, separated by the word “or”.

    c. A sentence that ends in a period.

3. Write expressions to modify a string, `str` in each of the following ways.

a. Replace all instances of words that begin with a capital letter with “Someone”.

b. Convert the letter at the start of each word to a capital.

c. Reverse any two words separated by “or”. E.g., “this or that” should become “that or this”.

d. Drop any part of the string that comes before “Alice”.

e. Drop any part of the string that comes after “Rabbit”.

4. Write a procedure, `(count-alphabetically-first strings)`, that takes a list of strings as input, identifies the alphabetically first string in the list, and returns a count of the number of times that string appears in the list.

```
> (count-alphabetically-first '("some" "are" "short" "some" "are"
"quite" "long"))
      2     ; "are" is alphabetically first
> (count-alphabetically-first '("some" "are" "short" "and" "some" "are"
"long"))
      1     ; "and" is alphabetically first
```

```
(define count-alphabetically-first
```

Answer:

1. What are the results to the following expressions?

```
> (define example "my name is enumerable")
> (define lst (list "5" "2" "6" "0" "1"))
> (regexp-match* #px"n[^a]*m." example)
    '("nume")
> (string-split example #px"n.m")
    '("my " "e is e" "erable")
> (regexp-replace* #px"([a-z]*) ([a-z]*)" example "\\1,\\2,\\1")
    "my,name,my,is,,enumerable,"
> (map (section + <> (reduce + (list 1 2 3))) lst)
    Error: lst contains string element while + procedure
requires number
    expected: number?
    given: "5"
    argument position: 1st
> (map (o add1 string->number) lst)
    '(6 3 7 1 2)
```

2. Write regular expressions for each of the following.

a. Words that contain two vowels in sequence.

```
#px"\\w*[aeiouAEIOU][aeiouAEIOU]\\w*"
```

b. Two words, separated by the word “or”.

```
#px"[a-z]+ or [a-z]+"
```

c. A sentence that ends in a period.

```
#px"[A-Z][a-z ]*[.]"
```

3. Write expressions to modify a string, str in each of the following ways.

a. Replace all instances of words that begin with a capital letter with “Someone”.

```
(regexp-replace* #px"\\s[A-Z][\\w]+" str " Someone")
```

b. Convert the letter at the start of each word to a capital.

```
(regexp-replace* #px"([a-z])([a-zA-Z]+)" str
                 (lambda (all one two)
                   (string-append (string-upcase one)
                                  two)))
```

c. Reverse any two words separated by “or”. E.g., “this or that” should become “that or this”.

```
(regexp-replace* #px"([a-z]+) or ([a-z]+)" str "\\2 or
\\1")
```

d. Drop any part of the string that comes before “Alice”.

```
(regexp-replace* #px"(.*)(Alice)" str "\\2")
```

e. Drop any part of the string that comes after “Rabbit”.

```
(regexp-replace* #px"(Rabbit)(.*)" str "\\1")
```

4. Write a procedure, (count-alphabetically-first strings), that takes a list of strings as input, identifies the alphabetically first string in the list, and returns a count of the number of times that string appears in the list.

```
> (count-alphabetically-first `("some" "are" "short" "some" "are"
"quite" "long"))
2     ; "are" is alphabetically first
> (count-alphabetically-first `("some" "are" "short" "and" "some" "are"
"long"))
1     ; "and" is alphabetically first
```

```
(define count-alphabetically-first
  (lambda (strings)
    (tally-value strings
                 (list-ref (sort strings string-ci<?) 0))))
```