Practice Quiz (02/21/2019)

1. Consider the following list.

```
(define stuff (list 1 2.3 4 5/6 7+8i "nine" 'ten "11" 12.13+14i))
```

What do you expect as output for each of the following expressions?

```
a. (filter string? stuff)
```

```
b. (filter complex? stuff)
```

```
c. (filter (negate real?) stuff)
```

```
d. (filter integer? stuff)
```

```
e. (filter exact? stuff)
```

```
f. (filter (conjoin number? exact?) stuff)
```

```
g. (filter (conjoin number? (negate real?)) stuff)
```

3.
```
(define fun
  (lambda (val)
    (if (> val 15/4)
        "A"
        (if (> val 11/4)
            "B"
            (if (> val 7/4)
                "C"
```

```
                (if (> val 1)
                    "D"
                    "F"))))))
```

a. What is (fun 3.5)?

b. What is (fun 2)?

c. What does fun do?

d. Rewrite fun using cond.

3. Write a procedure, (add-indefinite-article str) that takes as input a string that starts with a lowercase letter and returns that string an indefinite article added.

```
> (add-indefinite-article "antelope")
"an antelope"
> (add-indefinite-article "bat")
"a bat"
> (add-indefinite-article "iguana")
"an iguana"
> (add-indefinite-article "jackalope")
"a jackalope"
```

4. Consider the following average procedure that we have encountered a few times of late.

```
(define average
      (lambda (vals)
            (/ (reduce + vals) (length vals))))
```

What preconditions does average have? Describe some tests for edge cases.

```
; 1
> (filter string? stuff)
'("nine" "11")
> (filter complex? stuff)
'(1 2.3 4 5/6 7+8i 12.13+14.0i) ; Tricky: Real numbers are also
counted as complex
> (filter (negate real?) stuff)
'(7+8i "nine" ten "11" 12.13+14.0i)
> (filter integer? stuff)
'(1 4)
> (filter exact? stuff)
. . exact?: contract violation ; Tricky: exact? can only take
numbers as arguments
  expected: number?
  given: "nine"
> (filter (conjoin number? exact?) stuff)
'(1 4 5/6 7+8i)
> (filter (conjoin number? (negate real?)) stuff)
'(7+8i 12.13+14.0i)

;2
; a. "B"
; b. "C"
; c. Calculates grades based on val
; d.

(define fun
  (lambda (val)
    (cond [(> val 15/4)
        "A"]
        [(> val 11/4)
            "B"]
            [(> val 7/4)
                "C"]
                [(> val 1)
                    "D"]
                [else
                    "F"])))

;3
(define add-indefinite-article
  (lambda (str)
    (if (equal? (string-ref str 0) (or #\a #\e #\i #\o #\u))
        (string-append "an " str)
        (string-append "a " str))))

;4
```

```
; Preconditions: vals must be nonempty, vals must be a list of
numbers
; Tests: different types of numbers (complex, real, integer,
rational), large lists, empty list, singleton list
```