CMPU 101 § 55 · Computer Science I · Spring 2019

Assignment 3

Due February 27, 9:00 a.m.

On this assignment, you will write a few predicates, i.e., functions that return true or false. For full credit, you should only use `and`, `or`, and `not`, not the `if` and `cond` conditions. If you find this difficult to think about, you may want to begin by writing a version which uses `if` or `cond` and then work towards one which does not.

Problem 1

Write a (non-recursive) Scheme predicate called `leap-year?` that takes as its only input a number representing a year between 1901 and 2099. Use the following rules to determine whether the input year is a leap year:

- Any year that is not evenly divisible by 4 is not a leap year.
- Any year that is evenly divisible by 4 is a leap year *unless* the rule that follows applies.
- Any year that is evenly divisible by 100 is not a leap year *unless* the rule that follows applies.
- Any year that is evenly divisible by 400 is a leap year.

You may want to define a "helper function" to test whether one number divides evenly by another, which can be done by checking its remainder.

Write at least four test cases, being careful to test that all the conditions are handled correctly, e.g., don't just odd numbered years!

Problem 2

That problem was hard work. You're hungry. You'd like some…fruit? But you're surrounded by lists and you don't know which of them contain fruit. Scheme to the rescue! Write a recursive predicate `any-fruit?` that checks whether a list contains delicious fruit. Your predicate should stop checking the list as soon as it finds fruit.

```
(any-fruit? '())                      ⇒  #f
(any-fruit? '(apple))                 ⇒  #t
(any-fruit? '(hunger sadness orange))  ⇒  #t
```

Hint: You may want to define the predicate `fruit?` to use as a helper function that checks if its input is a kind fruit you've heard of.

Problem 3

Write a recursive predicate `all-fruit?` that takes a list of any length and returns `#t` if the list consists entirely of fruit and `#f` if it's contaminated by things that are not fruit.

```
(all-fruit? '())                          ⇒  #t
(all-fruit? '(apple))                     ⇒  #t
(all-fruit? '(apple pear broccoli cat))   ⇒  #f
```

You can re-use your `fruit?` predicate from Problem 1.

Problem 4

We are concerned that our students are eating too much fruit. Therefore, each list should have only *one* piece of fruit in it. Write a recursive predicate `just-one-piece?` that takes a list of any size as input and returns `#t` if there is one and only one piece of fruit in the list.

Your solution should call the `any-fruit?` predicate you wrote above. For `just-one-piece?`, write at least three test cases.

Submitting

Don't forget to submit your work using the `submit101` command!

```
submit101 g-asmt03 asmt03
```

(If the name of your directory is different from `asmt03`, change `asmt03` to whatever the name of your directory is.)