```
# CMPT 145 Course material
# Copyright (c) 2017-2021 Michael C Horsch
# All rights reserved.
#
# This document contains resources for homework assigned to students of
# CMPT 145 and shall not be distributed without permission.  Posting this
# file to a public or private website, or providing this file to a person
# not registered in CMPT 145, constitutes Academic Misconduct, according
# to the University of Saskatchewan Policy on Academic Misconduct.
#
# Synopsis:
#    Assignment 7 solutions

import a7q6 as a7q6

# one string used to report any error
# format:  "<test item>: <reason>; found '<result>', expected '<expected>'"
assert_report = "{}: {}; found '{}', expected '{}'"


###### testing recsumlist() #####
#### case
test_item = "recsumlisti()"
reason = 'base case 0'
input = []
expected = 0
result = a7q6.recsumlisti(0, input)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case 1'
input = [1.01]
expected = sum(input)
result = a7q6.recsumlisti(0, input)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case 2'
input = [1.01, 2.02]
expected = sum(input)
result = a7q6.recsumlisti(0, input)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

###### testing recmemberlisti() #####
#### case
test_item = "recmemberlisti()"
reason = 'base case 1'
input_list = []
input_value = 1
expected = (input_value in input_list)
result = a7q6.recmemberlisti(0, input_value, input_list)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'base case 2'
```

```
input_list = [2]
input_value = input_list[0]
expected = (input_value in input_list)
result = a7q6.recmemberlisti(0, input_value, input_list)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case: first'
input_list = [1, 3, 5, 7, 2, 4, 6, 9]
input_value = 1
expected = (input_value in input_list)
result = a7q6.recmemberlisti(0, input_value, input_list)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case: last'
input_list = [1, 3, 5, 7, 2, 4, 6, 9]
input_value = 9
expected = (input_value in input_list)
result = a7q6.recmemberlisti(0, input_value, input_list)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case: middle'
input_list = [1, 3, 5, 7, 2, 4, 6, 9]
input_value = 7
expected = (input_value in input_list)
result = a7q6.recmemberlisti(0, input_value, input_list)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case: not member'
input_list = [1, 3, 5, 7, 2, 4, 6, 9]
input_value = 8
expected = (input_value in input_list)
result = a7q6.recmemberlisti(0, input_value, input_list)
assert result == expected, assert_report.format(test_item, reason, result,
expected)


###### testing reccountlisti() #####
#### case
test_item = "reccountlisti()"
reason = 'base case 1'
input_list = []
input_value = 1
expected = 0
result = a7q6.reccountlisti(0, input_value, input_list)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'base case 2'
input_list = [2]
input_value = input_list[0]
```

```
expected = 1
result = a7q6.reccountlisti(0, input_value, input_list)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case: first'
input_list = [1, 2, 3, 4, 1, 2, 3]
input_value = 1
expected = 2
result = a7q6.reccountlisti(0, input_value, input_list)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case: last'
input_list = [1, 2, 3, 4, 3, 1, 2, 3]
input_value = 3
expected = 3
result = a7q6.reccountlisti(0, input_value, input_list)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case:  middle'
input_list = [1, 2, 3, 4, 1, 2, 3]
input_value = 4
expected = 1
result = a7q6.reccountlisti(0, input_value, input_list)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case:  not present'
input_list = [1, 2, 3, 4, 1, 2, 3]
input_value = 5
expected = 0
result = a7q6.reccountlisti(0, input_value, input_list)
assert result == expected, assert_report.format(test_item, reason, result,
expected)
```