```
# 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 a7q7 as a7q7
import node as N

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


###### testing recsumlist() #####
#### case
test_item = "sumnc_rec()"
reason = 'base case 0'
input = None
expected = 0
result = a7q7.sumnc_rec(input)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case 1'
input = N.node(1)
expected = 1
result = a7q7.sumnc_rec(input)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case 2'
input = N.node(1, N.node(2))
expected = 3
result = a7q7.sumnc_rec(input)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

###### testing membernc_rec() #####
#### case
test_item = "membernc_rec()"
reason = 'base case 1'
input_chain = None
input_value = 1
expected = False
result = a7q7.membernc_rec(input_value, input_chain)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
```

```
reason = 'base case 2'
input_chain = N.node(1)
input_value = input_chain.data
expected = True
result = a7q7.membernc_rec(input_value, input_chain)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case: first'
input_chain = N.node(1, N.node(3, N.node(5, N.node(7, N.node(2, N.node(4, N.node(6,
N.node(9))))))))
input_value = 1
expected = True
result = a7q7.membernc_rec(input_value, input_chain)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case: last'
input_chain = N.node(1, N.node(3, N.node(5, N.node(7, N.node(2, N.node(4, N.node(6,
N.node(9))))))))
input_value = 9
expected = True
result = a7q7.membernc_rec(input_value, input_chain)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case: middle'
input_chain = N.node(1, N.node(3, N.node(5, N.node(7, N.node(2, N.node(4, N.node(6,
N.node(9))))))))
input_value = 7
expected = True
result = a7q7.membernc_rec(input_value, input_chain)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case: not member'
input_chain = N.node(1, N.node(3, N.node(5, N.node(7, N.node(2, N.node(4, N.node(6,
N.node(9))))))))
input_value = 8
expected = False
result = a7q7.membernc_rec(input_value, input_chain)
assert result == expected, assert_report.format(test_item, reason, result,
expected)


###### testing countnc_rec() #####
#### case
test_item = "countnc_rec()"
reason = 'base case 1'
input_chain = None
input_value = 1
expected = 0
result = a7q7.countnc_rec(input_value, input_chain)
assert result == expected, assert_report.format(test_item, reason, result,
expected)
```

```
#### case
reason = 'base case 2'
input_chain = N.node(2)
input_value = input_chain.data
expected = 1
result = a7q7.countnc_rec(input_value, input_chain)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case: first'
input_chain = N.node(1, N.node(2, N.node(3, N.node(4, N.node(1, N.node(2,
N.node(3)))))))
input_value = 1
expected = 2
result = a7q7.countnc_rec(input_value, input_chain)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case: last'
input_chain = N.node(1, N.node(2, N.node(3, N.node(4, N.node(3, N.node(1, N.node(2,
N.node(3))))))))
input_value = 3
expected = 3
result = a7q7.countnc_rec(input_value, input_chain)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case:  middle'
input_chain = N.node(1, N.node(2, N.node(3, N.node(4, N.node(1, N.node(2,
N.node(3)))))))
input_value = 4
expected = 1
result = a7q7.countnc_rec(input_value, input_chain)
assert result == expected, assert_report.format(test_item, reason, result,
expected)

#### case
reason = 'recursive case:  not present'
input_chain = N.node(1, N.node(2, N.node(3, N.node(4, N.node(1, N.node(2,
N.node(3)))))))
input_value = 5
expected = 0
result = a7q7.countnc_rec(input_value, input_chain)
assert result == expected, assert_report.format(test_item, reason, result,
expected)
```