python

import unittest

from unittest.mock import patch

from io import StringIO

import sys

class TestGameFunctions(unittest.TestCase):

def test\_computer\_shoots(self):

# Testing shooting at all available blocks

test\_set = {(16, 1), (16, 2), (17, 1)}

for \_ in range(len(test\_set)):

result = computer\_shoots(test\_set)

self.assertIn(result, test\_set)

test\_set.discard(result)

self.assertEqual(len(test\_set), 0)

def test\_check\_hit\_or\_miss\_hit(self):

# Testing hit scenario

opponents\_ships\_list = [[(16, 1)], [(17, 1)]]

opponents\_ships\_set = {(16, 1), (17, 1)}

fired\_block = (16, 1)

self.assertTrue(check\_hit\_or\_miss(fired\_block, opponents\_ships\_list, False, opponents\_ships\_list, opponents\_ships\_set))

def test\_check\_hit\_or\_miss\_miss(self):

# Testing miss scenario

opponents\_ships\_list = [[(16, 1)], [(17, 1)]]

opponents\_ships\_set = {(16, 1), (17, 1)}

fired\_block = (18, 1)

self.assertFalse(check\_hit\_or\_miss(fired\_block, opponents\_ships\_list, False, opponents\_ships\_list, opponents\_ships\_set))

@patch('sys.stdout', new\_callable=StringIO)

def test\_show\_message\_at\_rect\_center(self, mock\_stdout):

# Testing message display

message = "Test message"

rect = (0, 0, 100, 50)

show\_message\_at\_rect\_center(message, rect)

self.assertIn(message, mock\_stdout.getvalue())

def test\_ship\_is\_valid(self):

# Ship validation

ship\_set = {(16, 1), (16, 2)}

blocks\_for\_manual\_drawing = {(16, 3)}

self.assertTrue(ship\_is\_valid(ship\_set, blocks\_for\_manual\_drawing))

def test\_ship\_is\_not\_valid(self):

# Ship not valid (overlap)

ship\_set = {(16, 1), (16, 2)}

blocks\_for\_manual\_drawing = {(16, 1), (16, 3)}

self.assertFalse(ship\_is\_valid(ship\_set, blocks\_for\_manual\_drawing))

def test\_draw\_ships(self):

# Placeholder test for drawing

ships\_coordinates\_list = [[(16, 1), (16, 2)]]

draw\_ships(ships\_coordinates\_list) # Draws on screen, nothing to assert

def test\_draw\_from\_dotted\_set(self):

# Placeholder test for drawing

draw\_from\_dotted\_set({(16, 1), (16, 2)}) # Draws on screen, nothing to assert

def test\_draw\_hit\_blocks(self):

# Placeholder test for drawing

draw\_hit\_blocks({(16, 1), (16, 2)}) # Draws on screen, nothing to assert

def test\_add\_missed\_block\_to\_dotted\_set(self):

# Testing block is added to dotted set

dotted\_set.clear()

fired\_block = (16, 1)

add\_missed\_block\_to\_dotted\_set(fired\_block)

self.assertIn(fired\_block, dotted\_set)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()