7) Program to implement monkey banana problem.

```python
import unittest
import logging


logging.basicConfig(level=logging.INFO, format='%(levelname)s - %(message)s')

class MonkeyBananaProblem:
    def __init__(self):
        self.monkey_position = 'floor'
        self.banana_position = 'high'
        self.stool_position = 'floor'
        self.actions = []

    def perform_action(self, action):
        if action == 'move_to_stool':
            if self.monkey_position == 'floor' and self.stool_position == 'floor':
                self.monkey_position = 'stool'
                self.actions.append(action)
            else:
                return False
        elif action == 'climb':
            if self.monkey_position == 'stool' and self.banana_position == 'high':
                self.monkey_position = 'high'
                self.actions.append(action)
            else:
                return False
        elif action == 'grab_banana':
            if self.monkey_position == 'high' and self.banana_position == 'high':
                self.banana_position = 'held'
                self.actions.append(action)
            else:
                return False
        elif action == 'descend':
            if self.monkey_position == 'high':
                self.monkey_position = 'stool'
                self.actions.append(action)
            else:
                return False
        elif action == 'move_to_floor':
            if self.monkey_position == 'stool':
                self.monkey_position = 'floor'
                self.actions.append(action)
            else:
                return False
        else:
            return False
        return True

    def solve(self):
        actions = [
            'move_to_stool',
            'climb',
            'grab_banana',
            'descend',
            'move_to_floor'
        ]
        for action in actions:
            if not self.perform_action(action):
                return "Failed to perform action: " + action
        return "Banana acquired!"

class TestMonkeyBananaProblem(unittest.TestCase):
    def setUp(self):
        self.problem = MonkeyBananaProblem()

    def test_initial_state(self):
        self.assertEqual(self.problem.monkey_position, 'floor')
        self.assertEqual(self.problem.banana_position, 'high')
        self.assertEqual(self.problem.stool_position, 'floor')
        self.assertEqual(self.problem.actions, [])
        logging.info("Initial state test passed.")

    def test_actions(self):
        self.assertTrue(self.problem.perform_action('move_to_stool'))
        self.assertEqual(self.problem.monkey_position, 'stool')
        self.assertTrue(self.problem.perform_action('climb'))
        self.assertEqual(self.problem.monkey_position, 'high')
        self.assertTrue(self.problem.perform_action('grab_banana'))
        self.assertEqual(self.problem.banana_position, 'held')
        self.assertTrue(self.problem.perform_action('descend'))
        self.assertEqual(self.problem.monkey_position, 'stool')
        self.assertTrue(self.problem.perform_action('move_to_floor'))
        self.assertEqual(self.problem.monkey_position, 'floor')
        logging.info("Actions test passed.")

    def test_solve(self):
        result = self.problem.solve()
        self.assertEqual(result, "Banana acquired!")
        self.assertEqual(self.problem.actions, [
            'move_to_stool',
            'climb',
            'grab_banana',
            'descend',
            'move_to_floor'
        ])
        logging.info("Solve test passed.")

    def test_invalid_action(self):
        result = self.problem.perform_action('invalid_action')
        self.assertFalse(result)
        logging.info("Invalid action test passed.")

    def test_failed_action(self):

        self.problem.monkey_position = 'floor'
        result = self.problem.perform_action('climb')
        self.assertFalse(result)
        logging.info("Failed action test result: %s", result)

    def tearDown(self):

        logging.info("Actions taken: %s", self.problem.actions)

if __name__ == "__main__":
    unittest.main()
```