Unit Testing 3 - Solutions

1. We need to create a `Mock` object with a return value that we can pass to the actual function.

```
import random
from unittest.mock import Mock

def get_random_number():
    return random.randint(1, 10)

def add_number_with_random_number(a, get_random_number):
    return a + get_random_number()

def test_add_number_with_random_number():
    # Arrange
    mock_get_random_number = Mock()
    mock_get_random_number.return_value = 5
    a = 5
    expected = 10

    # Act
    actual = add_number_with_random_number(a, mock_get_random_number)

    # Assert
    assert expected == actual

test_add_number_with_random_number()
```

2. Same principles as question 1 apply here.

```
from random import randint
from unittest.mock import Mock

def get_random_number(randint):
    return randint(1, 10)



def test_get_random_number():
    # Arrange
    mock_randint = Mock()
    mock_randint.return_value = 5
    expected = 5

    # Act
    actual = get_random_number(mock_randint)

    # Assert
    assert expected == actual
```

3.

```python
from unittest.mock import patch

def get_user_details():
    name = input('Please enter your name: ')
    age = int(input('Please enter your age: '))
    print(f'Thank you, your name is {name} and your age is {age}')

@patch('builtins.input', side_effect=['Jane', 25])
@patch('builtins.print')
def test_get_user_details(mock_print, mock_input):
    get_user_details()

    mock_print.assert_called_with("Thank you, your name is Jane and your age is 25")
    assert mock_input.call_count == 2
    assert mock_print.call_count == 1
```