shoppingcart.py

```
class UnAvailableItem(Exception):
    pass


class IndexException(Exception):
    pass


class Shoppingcart(object):
    cart = {}

    def __init__(self, item_list):
        self.item_list = item_list

    def add_to_cart(self, name, quantity):
        if name not in self.item_list:
            raise UnAvailableItem('The item {} is unavailable'.format(name))
        if not isinstance(quantity, int) or quantity < 1:
            raise IndexException('Quantity must be a positive integer')
        if name in self.cart:
            self.cart[name] = self.cart[name] + quantity
        else:
            self.cart[name] = quantity

    def update_to_cart(self, name, new_quantity):
        if name not in self.cart:
            raise UnAvailableItem('The item {} is not yet added to cart'.format(name))
        elif new_quantity == 0:
            del self.cart[name]
        elif not isinstance(new_quantity, int):
            raise IndexException('Invalid quantity')
        else:
            self.cart[name] = new_quantity

    def remove_from_cart(self, name):
        if name not in self.cart:
            raise UnAvailableItem('The item {} is not available in cart'.format(name))
        del self.cart[name]

    def print_bill(self):
        if not self.cart:
            raise UnAvailableItem('The cart is empty')
        else:
            bill = ""
            SI_NO = 1
            bill += f"{'SI No':<10} {'Item Name':>10} {'Quantity':>10} {'Price':>10}\n"
            bill += f"{'-----':<10} {'---------':>10} {'--------':>10} {'-----':>10}\n"
            for key in self.cart:
                for item in self.item_list:
                    if key == item:
                        total_price = int(self.item_list[item]) * self.cart[key]
                        bill += f"{SI_NO:<10} {key:>10} {self.cart[key]:>10} {total_price:>10.2f}\n"
```

```
                SI_NO += 1
                break
        return bill
```

shoppingcart_test.py

```
import pytest
from shoppingcart import Shoppingcart, UnAvailableItem, IndexException


@pytest.fixture
def shoppingcart():
    items_list = {"Milk": 28.50, "Apple": 72.00,
            "Cheese": 128.50, "Cream": 94.00,
            "Mango": 41.50, "Orange": 41.60}
    return Shoppingcart(items_list)

#ADDING
def test_add_item_to_cart(shoppingcart):
    shoppingcart.add_to_cart("Milk", 2)
    assert shoppingcart.cart['Milk'] == 2

def test_add_already_existing_item_to_cart(shoppingcart):
    shoppingcart.add_to_cart("Milk", 3)
    assert shoppingcart.cart['Milk'] == 5

#UPDATING
def test_update_cart_item(shoppingcart):
    shoppingcart.update_to_cart('Milk', 5)
    assert shoppingcart.cart['Milk'] == 5

def test_update_quantity_zero_delete_item(shoppingcart):
    shoppingcart.add_to_cart('Cheese', 2)
    shoppingcart.update_to_cart('Cheese',0)
    shoppingcart.remove_from_cart('Milk')
    assert len(shoppingcart.cart) == 0

#DELETING
def test_delete_cart_item(shoppingcart):
    shoppingcart.add_to_cart('Milk',5)
    shoppingcart.remove_from_cart('Milk')
    assert len(shoppingcart.cart) == 0

#PRINTBILL
def test_print_bill(shoppingcart):
    shoppingcart.add_to_cart("Cream", 3)
    shoppingcart.add_to_cart("Apple", 1)

    actual_output = shoppingcart.print_bill()
    assert "282.00" in actual_output.split()

#ADDING_NON_LISTED_ITEM
```

```
def test_add_notavailable_item_raises_exception(shoppingcart):
    with pytest.raises(UnAvailableItem):
        shoppingcart.add_to_cart('Fish', 1)

#ADDING WITH NON INTEGER QUANTITY
def test_add_non_integer_quantity_raises_exception(shoppingcart):
    with pytest.raises(IndexException):
        shoppingcart.add_to_cart('Milk', 2.4)

#UPDATE WITH NON AVAILABLE ITEM IN THE CART
def test_update_notavailable_item_raises_exception(shoppingcart):
    with pytest.raises(UnAvailableItem):
        shoppingcart.update_to_cart('Milk', 1)

#UPDATE WITH NON INTEGER QUANTITY
def test_update_non_interger_quantity_item_raises_exception(shoppingcart):
    with pytest.raises(IndexException):
        shoppingcart.add_to_cart('Milk',5)
        shoppingcart.update_to_cart('Milk', 1.4)

#DELETING NON AVAILABLE ITEM
def test_delete_notavailable_item_raises_exception(shoppingcart):
    with pytest.raises(UnAvailableItem):
        shoppingcart.add_to_cart('Milk',3)
        shoppingcart.remove_from_cart('Cream')
        shoppingcart.remove_from_cart('Cream')

#PRINT FOR EMPTY CART
def test_printbill_for_emptycart_raises_exception(shoppingcart):
    with pytest.raises(UnAvailableItem):
        shoppingcart.remove_from_cart('Milk')
        shoppingcart.remove_from_cart('Apple')
        shoppingcart.remove_from_cart('Orange')
        shoppingcart.print_bill()
```