```python
# Author: Michael Oliveri
# Email: mcoliveri@umass.edu
# Spire ID: 34881210

class VendingMachine:
    def __init__(self):
        self.items = {}
        self.balance = 0.0
        self.sales = 0.0
        self.sales_history = []

    def add_item(self, name, price, quantity):
        if name in self.items:
            current_price, current_quantity = self.items[name] # Used to unpack,
current_price is important even though greyed out. do not delete
            self.items[name] = (price, current_quantity + quantity)
        else:
            self.items[name] = (price, quantity)
        print(f"{quantity} {name}(s) added to inventory")

    def get_item_price(self, name):
        if name in self.items:
            return self.items[name][0]
        else:
            print("Invalid item")
            return None

    def get_item_quantity(self, name):
        if name in self.items:
            return self.items[name][1]
        else:
            print("Invalid item")
            return None

    def list_items(self):
        if not self.items:
            print("No items in the vending machine")
        else:
            print("Available items:")
            for item, (price, quantity) in sorted(self.items.items()):
                print(f"{item} (${price}): {quantity} available")
```

```
    def insert_money(self, amount):
        if amount in [1.0, 2.0, 5.0]:
            self.balance += amount
            self.balance = round(self.balance, 2)
            print(f"Balance: {self.balance}")
        else:
            print("Invalid amount")

    def purchase(self, name):
        if name not in self.items:
            print("Invalid item")
        elif self.items[name][1] == 0:
            print(f"Sorry {name} is out of stock")
        elif self.balance < self.items[name][0]:
            print(f"Insufficient balance. Price of {name} is {self.items[name][0]}")
        else:
            price = self.items[name][0]
            self.items[name] = (price, self.items[name][1] - 1)  # Reduce quantity
            self.balance -= price
            self.balance = round(self.balance, 2)
            self.sales += price
            self.sales = round(self.sales, 2)
            self.sales_history.append((name, price))
            print(f"Purchased {name}")
            print(f"Balance: {self.balance}")

    def output_change(self):
        if self.balance > 0:
            print(f"Change: {self.balance}")
            self.balance = 0.0
        else:
            print("No change")

    def remove_item(self, name):
        if name in self.items:
            del self.items[name]
            print(f"{name} removed from inventory")
        else:
            print("Invalid item")

    def empty_inventory(self):
        self.items.clear()
```

```
        print("Inventory cleared")

    def get_total_sales(self):
        return self.sales

    def stats(self, N):
        if not self.sales_history:
            print("No sale history in the vending machine")
            return

        recent_sales = self.sales_history[-N:]
        print(f"Sale history for the most recent {len(recent_sales)} purchase(s):")

        sale_data = {}
        for item, price in recent_sales:
            if item not in sale_data:
                sale_data[item] = {'total_sales': 0, 'count': 0}
            sale_data[item]['total_sales'] += price
            sale_data[item]['count'] += 1

        for item in sorted(sale_data.keys()):
            total_sales = round(sale_data[item]['total_sales'], 2)
            count = sale_data[item]['count']
            print(f"{item}: ${total_sales} for {count} purchase(s)")
```