UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 4

Arrays

Submitted by:
Enverzo, Kyle Andrey D.

Instructor:
Engr. Maria Rizette H. Sayo

August, 16, 2025

I. Objectives

Introduction

Array, in general, refers to an orderly arrangement of data elements. Array is a type of data structure that stores data elements in adjacent locations. Array is considered as linear data structure that stores elements of same data types. Hence, it is also called as a linear homogenous data structure.

This laboratory activity aims to implement the principles and techniques in:

- Writing algorithms using Array data structure
- Solve programming problems using dynamic memory allocation, arrays and pointers

II. Methods

<u>Jenna's Grocery</u>

Jenna's Grocery List		
Apple	PHP 10	x7
Banana	PHP 10	x8
Broccoli	PHP 60	x12
Lettuce	PHP 50	x10

Jenna wants to buy the following fruits and vegetables for her daily consumption. However, she needs to distinguish between fruit and vegetable, as well as calculate the sum of prices that she has to pay in total.

Problem 1: Create a class for the fruit and the vegetable classes. Each class must have a constructor, deconstructor, copy constructor and copy assignment operator. They must also have all relevant attributes (such as name, price and quantity) and functions (such as calculate sum) as presented in the problem description above.

Problem 2: Create an array GroceryList in the driver code that will contain all items in Jenna's Grocery List. You must then access each saved instance and display all details about the items.

Problem 3: Create a function TotalSum that will calculate the sum of all objects listed in Jenna's Grocery List.

Problem 4: Delete the Lettuce from Jenna's GroceryList list and de-allocate the memory assigned.

III. Results

Present the visualized procedures done. Also present the results with corresponding data visualizations such as graphs, charts, tables, or image . Please provide insights, commentaries, or explanations regarding the data. If an explanation requires the support of literature such as academic journals, books, magazines, reports, or web articles please cite and reference them using the IEEE format.

Please take note of the styles on the style ribbon as these would serve as the style format of this laboratory report. The body style is Times New Roman size 12, line spacing: 1.5. Body text should be in Justified alignment, while captions should be center-aligned. Images should be readable and include captions. Please refer to the sample below:

ALGORITHM

1. Start
2. Define Classes:
    - Create GroceryItem base class with attributes (name, price, quantity) and methods (constructor, destructor, copy, calculate_sum, show)
    - Create Fruit and Vegetable subclasses inheriting from GroceryItem
3. Initialize Shopping Cart:
    - Create a list containing instances of Fruit and Vegetable with their respective details
4. Display Grocery List:
    - Iterate through the shopping cart and display each item's details using the show() method
5. Calculate Total Cost:
    - Sum up the total cost of all items using calculate_sum() method
6. Modify Shopping Cart:
    - Remove "Lettuce" from the shopping cart
    - Display the updated list and new total cost
7. End

FLOWCHART

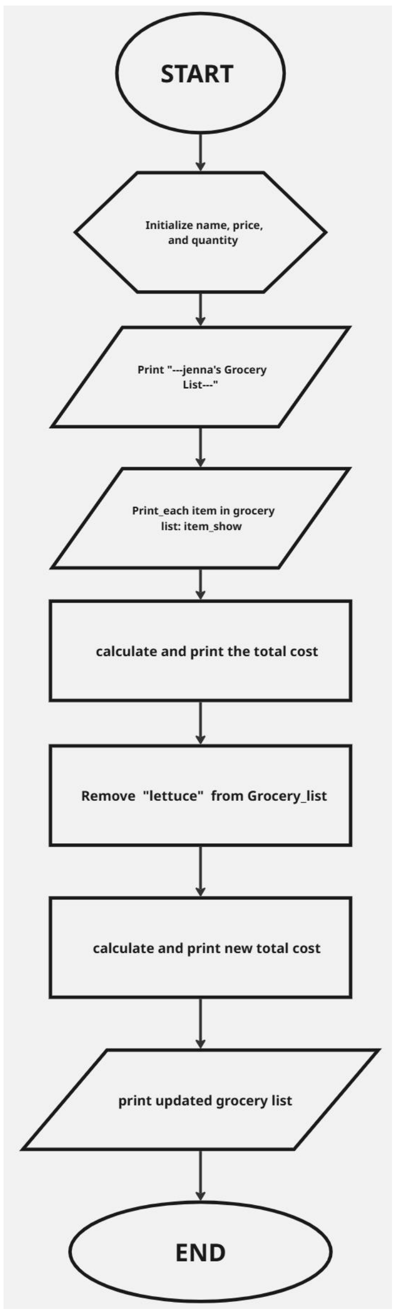
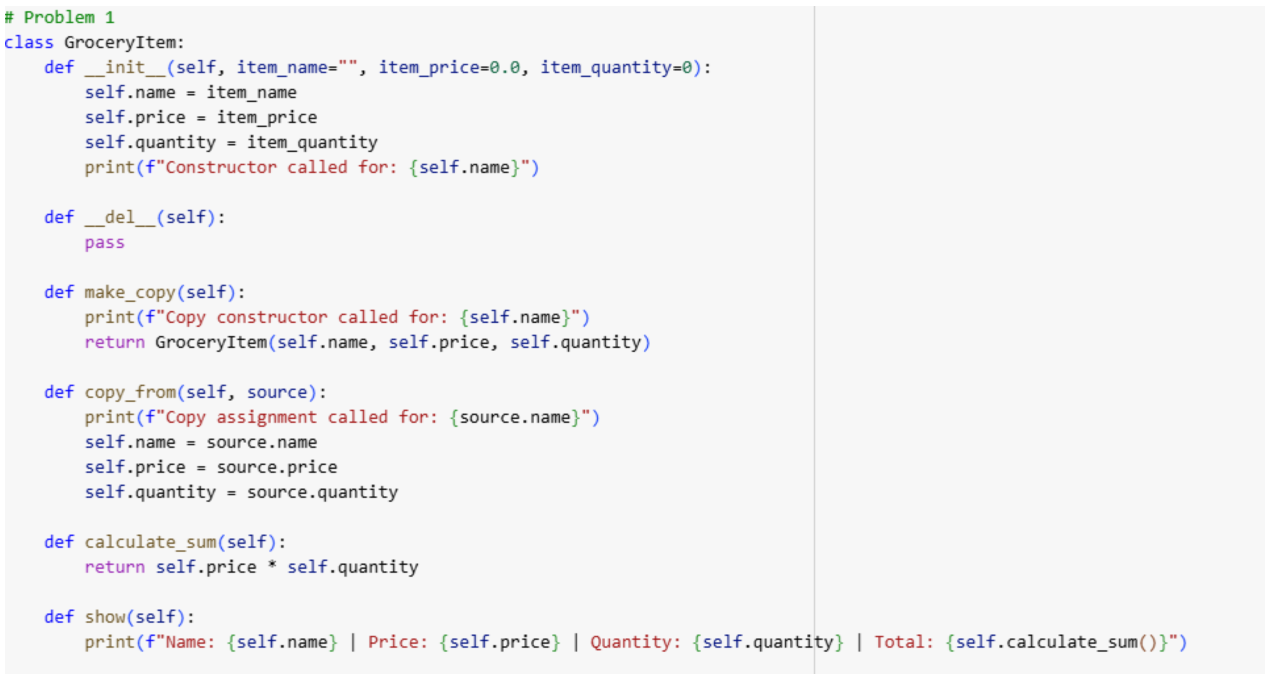

START
Initialize name, price, and quantity
Print "---jenna's Grocery List---"
Print_each item in grocery list: item_show
calculate and print the total cost
Remove "lettuce" from Grocery_list
calculate and print new total cost
print updated grocery list
END

Figure 1 Flowchart

This flowchart shows the program's workflow: (1) Initialize shopping cart with items, (2) Display each item's details in a loop, (3) Calculate total cost, (4) Remove "Lettuce", (5) Show updated list and new total. It uses standard symbols - ovals (start/end), rectangles (processes), parallelograms (input/output), and diamonds (decisions) - to visualize the code's logic from start to finish.

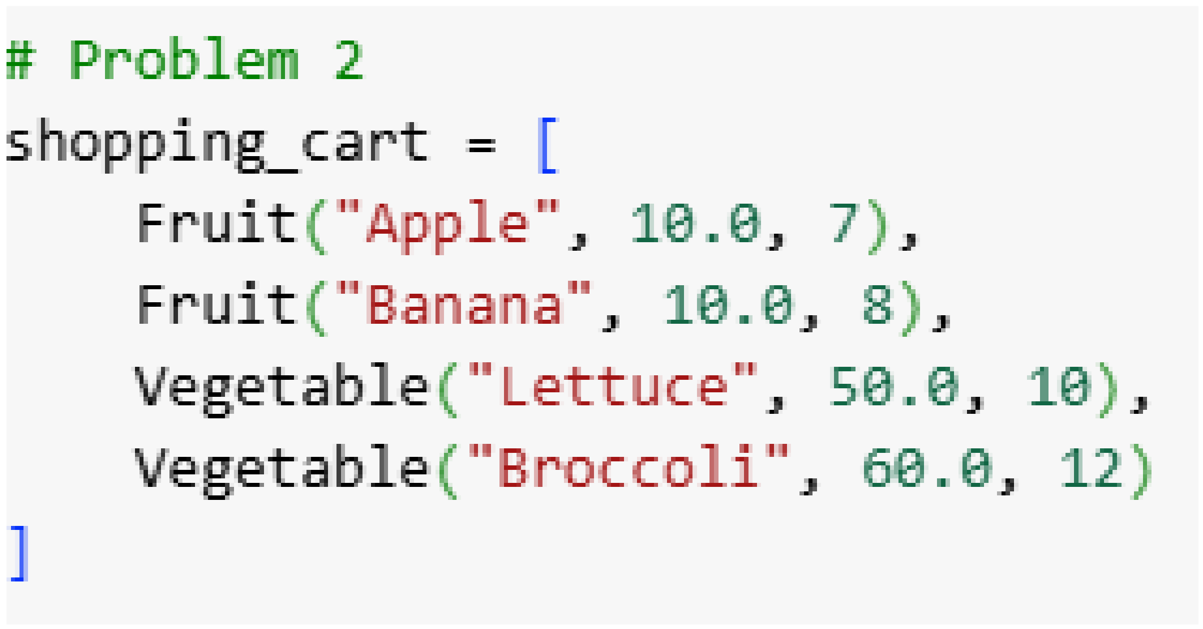
PROGRAM



```
# Problem 1
class GroceryItem:
    def __init__(self, item_name="", item_price=0.0, item_quantity=0):
        self.name = item_name
        self.price = item_price
        self.quantity = item_quantity
        print(f"Constructor called for: {self.name}")

    def __del__(self):
        pass

    def make_copy(self):
        print(f"Copy constructor called for: {self.name}")
        return GroceryItem(self.name, self.price, self.quantity)

    def copy_from(self, source):
        print(f"Copy assignment called for: {source.name}")
        self.name = source.name
        self.price = source.price
        self.quantity = source.quantity

    def calculate_sum(self):
        return self.price * self.quantity

    def show(self):
        print(f"Name: {self.name} | Price: {self.price} | Quantity: {self.quantity} | Total: {self.calculate_sum()}")
```

Figure 2 Program 1

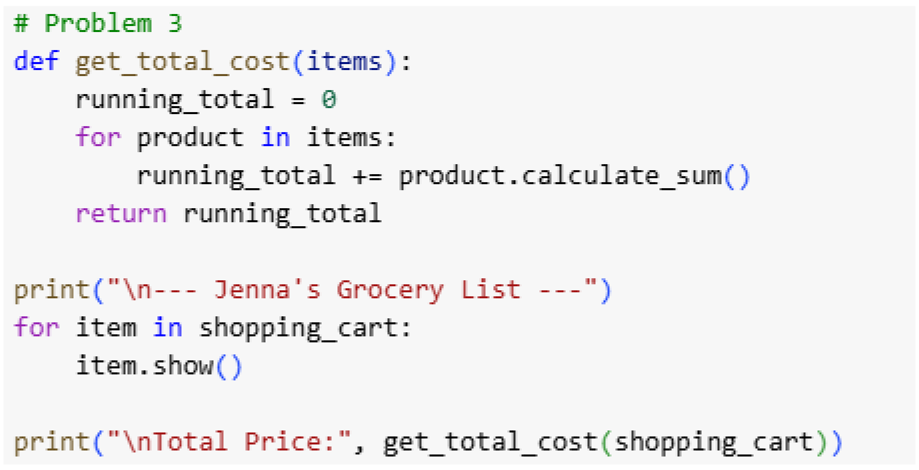
Builds the GroceryItem class to represent things with a name, quantity, and price. It has features to display information, figure out the total cost, duplicate the item, and print a notification when it is removed. Fruit and Vegetable, two additional classes, inherit these characteristics without introducing any new ones.



```
# Problem 2
shopping_cart = [
    Fruit("Apple", 10.0, 7),
    Fruit("Banana", 10.0, 8),
    Vegetable("Lettuce", 50.0, 10),
    Vegetable("Broccoli", 60.0, 12)
]
```

Figure 3 Program 2

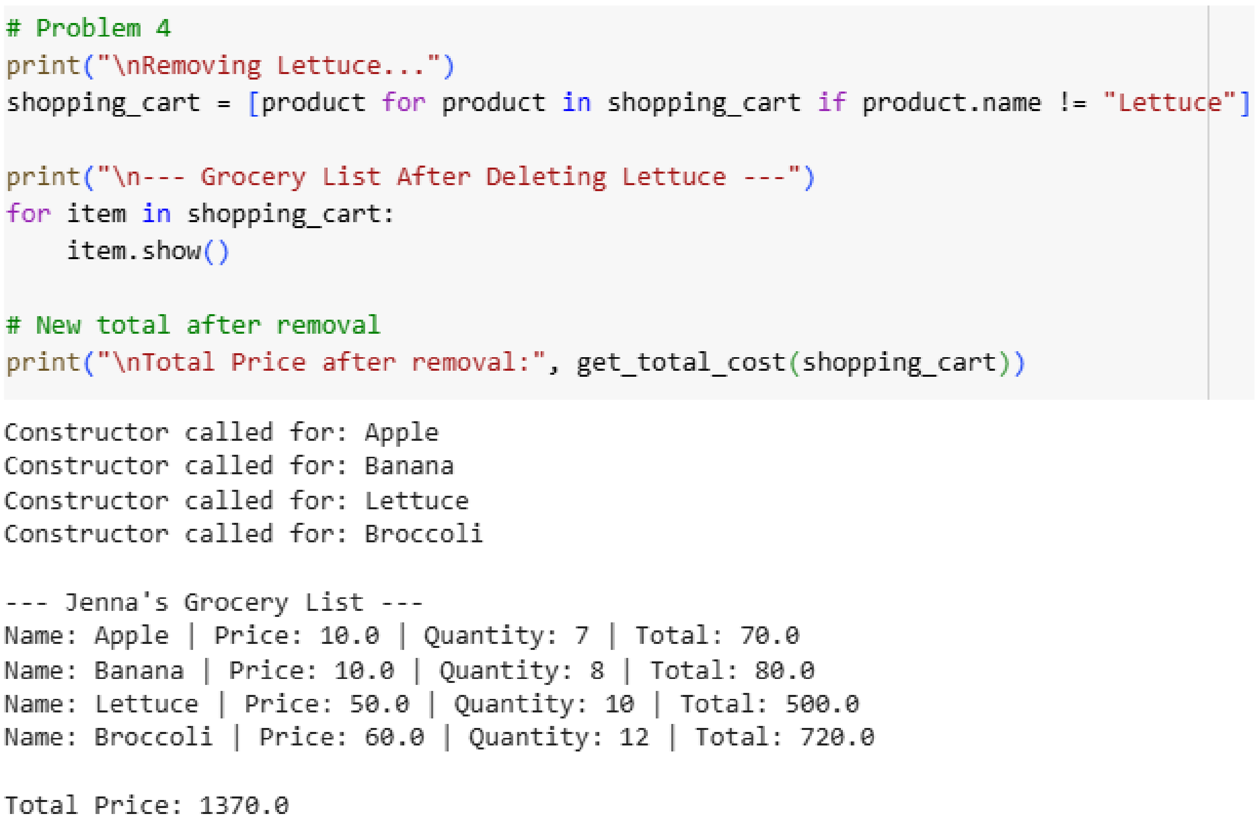
Uses the classes from Problem 1 to create a list of groceries, such as apples and broccoli. The name, cost, quantity, and price of each item are printed. Additionally, it employs a loop to present all the details and displays messages when objects are erased.



```python
# Problem 3
def get_total_cost(items):
    running_total = 0
    for product in items:
        running_total += product.calculate_sum()
    return running_total

print("\n--- Jenna's Grocery List ---")
for item in shopping_cart:
    item.show()

print("\nTotal Price:", get_total_cost(shopping_cart))
```

Figure 4 Program 3

Defines a function to figure out how much each item on the grocery list will cost overall. This demonstrates how to utilize functions to solve problems efficiently and neatly by using a loop to add up the sum of each item and printing the result: "Total Sum: PHP 1370."



```python
# Problem 4
print("\nRemoving Lettuce...")
shopping_cart = [product for product in shopping_cart if product.name != "Lettuce"]

print("\n--- Grocery List After Deleting Lettuce ---")
for item in shopping_cart:
    item.show()

# New total after removal
print("\nTotal Price after removal:", get_total_cost(shopping_cart))
```

```
Constructor called for: Apple
Constructor called for: Banana
Constructor called for: Lettuce
Constructor called for: Broccoli

--- Jenna's Grocery List ---
Name: Apple | Price: 10.0 | Quantity: 7 | Total: 70.0
Name: Banana | Price: 10.0 | Quantity: 8 | Total: 80.0
Name: Lettuce | Price: 50.0 | Quantity: 10 | Total: 500.0
Name: Broccoli | Price: 60.0 | Quantity: 12 | Total: 720.0

Total Price: 1370.0
```

Figure 5 Program 4

How to take a product off of a shopping list. Here, a filter is used to remove the item with the name "Lettuce." The code then prints the modified list with just the Apple, Banana, and Broccoli entries left. Additionally, a statement stating that lettuce has been removed is printed. Lastly, the updated total cost of PHP 870 is calculated and displayed. This section explains how to purge a list of particular items and update the outcomes.

IV. Conclusion

In summary, this grocery list program demonstrates object-oriented programming principles through the creation of GroceryItem, Fruit, and Vegetable classes. The algorithm efficiently manages a shopping cart by adding, displaying, and filtering items while calculating totals. The accompanying flowchart visually captures this logic—from initialization to output—using standard symbols to represent processes, decisions, and data flow. Together, the code, algorithm, pseudocode, and flowchart provide a complete and clear understanding of how the program works, making it easy to modify or scale for future enhancemen

References

[1] Python Software Foundation, "Python 3.12.0 Documentation," 2023. [Online]. Available: https://docs.python.org/3/.

[2] M. Lutz, *Learning Python*, 5th ed. Sebastopol, CA: O’Reilly Media, 2013.

[3] R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*. Upper Saddle River, NJ: Prentice Hall, 2008.

[4] S. N. Chapman, *The Fundamentals of Production Planning and Control*. Upper Saddle River, NJ: Pearson, 2006.

[5] IEEE, "IEEE Standard for Software Documentation (IEEE Std 1063-2021)," 2021. [Online]. Available: https://standards.ieee.org/standard/1063-2021.html.