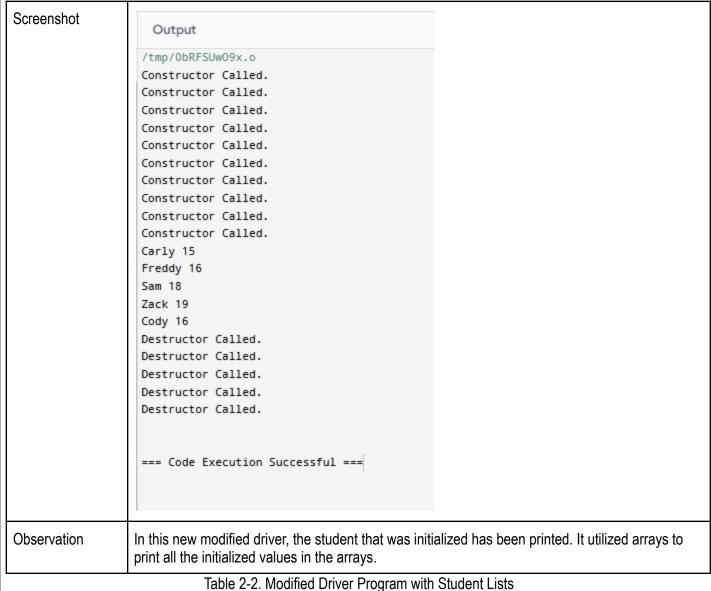
Hands-on Activity 2.1 Arrays, Pointers and Dynamic Memory Allocation

Course Code: CPE010	Program: Computer Engineering
Course Title: Data Structures and Algorithms	Date Performed: 11/09 24
Section:CPE21S4	Date Submitted: 11/09/24
Name(s): Pornobe, Reuel Christian, M.	Instructor: Prof. Maria Rizette Sayo

6. Output

Screenshot	Output		
	/tmp/SCk5z3Q8ch.o		
	Constructor Called.		
	Copy Constructor Called		
	Constructor Called.		
	Destructor Called.		
	Destructor Called.		
	Destructor Called.		
	=== Code Execution Successful ===		
Observation	In the first example of code, the initialization of the value of the variables were only intialized. It was not outputted by the program.		
Table 2-1. Initial Driver Program			



```
Loop A
Screenshot
                  for(int i = 0; i < j; i++){ //loop A
                  Student *ptr = new Student(namesList[i], ageList[i]);
                  studentList[i] = *ptr;
                  }
Loop B
Screenshot
                for(int i = 0; i < j; i++){
                    //loop B
                studentList[i].printDetails();
                }
```

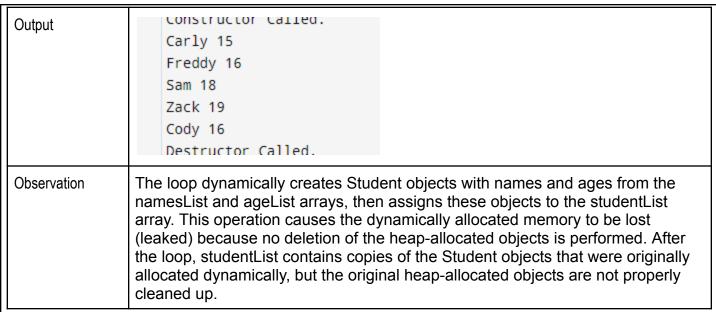


Table 2-3. Final Driver Program

```
Modifications
                  - int main() {
                        const size_t j = 5;
                        Student* studentList[j];
                        string namesList[j] = {"Carly", "Freddy", "Sam", "Zack", "Cody"};
                        int ageList[j] = {15, 16, 18, 19, 16};
                        for (size_t i = 0; i < j; i++) {
                            studentList[i] = new Student(namesList[i], ageList[i]);
                        }
                        for (size_t i = 0; i < j; i++) {
                            studentList[i]->printDetails();
                        }
                        // Cleanup: delete dynamically allocated Student objects
                        for (size_t i = 0; i < j; i++) {
                            delete studentList[i];
                        return 0;
                   }
Observation
                  Now the loop that Dynamically allocates objects are now managed properly and correctly because
```

all memory that was allocated is properly deallocated after using it.

7. Supplementary Activity

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 deallocate the memory assigned.

```
main.cpp
 1 #include <iostream>
 2 #include <string>
 3
 4 using namespace std;
 5
 6 // Problem 1: Create a class for the fruit and the vegetable classes.
 7 → class GroceryItem {
 8 public:
 9
        virtual ~GroceryItem() {} // Virtual destructor
        virtual double calculateTotal() const = 0;
11
        virtual void display() const = 0;
12 };
13
14 // Problem 1: Create a class for the fruit and the vegetable classes.
15 - class Fruit : public GroceryItem {
16 private:
        string itemName;
17
        double itemPrice;
18
        int itemQuantity;
19
20
21 public:
22
        // Constructor
23
        Fruit(const string& name, double price, int quantity)
            : itemName(name), itemPrice(price), itemQuantity(quantity) {}
24
        Fruit(const Fruit& other)
25
            : itemName(other.itemName), itemPrice(other.itemPrice), itemQuantity(other
26
                .itemQuantity) {}
27 -
        Fruit& operator=(const Fruit& other) {
28 -
            if (this != &other) {
                itemName = other.itemName;
29
30
               itemPrice = other.itemPrice;
               itemQuantity = other.itemQuantity;
31
32
           return *this;
33
34
35
        // Destructor
36
37
        ~Fruit() override {}
```

```
[] 6
                                                                                     ∝ Share
       main.cpp
        38
                // Problem 3: Create a function TotalSum that will calculate the sum of all object
R
        39
                    listed in Jenna's Grocery
        40 -
                double calculateTotal() const override {
        41
                    return itemPrice * itemQuantity;
        42
                }
5
        43 -
                void display() const override {
                    cout << "Fruit: " << itemName << ", Price: $" << itemPrice << ", Quantity: " <</pre>
        44
                        itemQuantity << endl;</pre>
        45
               }
        46 };
        47
        48 // Problem 1: Create a class for the fruit and the vegetable classes.
        49 - class Vegetable : public GroceryItem {
        50 private:
(3)
                string itemName;
        51
                double itemPrice;
        52
JS
        53
                int itemQuantity;
        54 public:
                // Constructor
        55
~GO
                Vegetable(const string& name, double price, int quantity)
        56
        57
                    : itemName(name), itemPrice(price), itemQuantity(quantity) {}
php
        58
                Vegetable(const Vegetable& other)
        59
                    : itemName(other.itemName), itemPrice(other.itemPrice), itemQuantity(other
                         .itemQuantity) {}
        60 +
                Vegetable& operator=(const Vegetable& other) {
R
        61 -
                    if (this != &other) {
                        itemName = other.itemName;
        62
                        itemPrice = other.itemPrice;
        63
                        itemQuantity = other.itemQuantity;
        64
        65
                    }
                    return *this;
        66
        67
                }
        68
                // Destructor
        69
        70
                ~Vegetable() override {}
        71
        72
                // Problem 3: Create a function TotalSum that will calculate the sum of all object
```

```
main.cpp
 71
 72
         // Problem 3: Create a function TotalSum that will calculate the sum of all object
            listed in Jenna's Grocery
 73 +
        double calculateTotal() const override {
 74
            return itemPrice * itemQuantity;
 75
 76
 77
        // Function to display details
 78 +
        void display() const override {
            cout << "Vegetable: " << itemName << ", Price: $" << itemPrice << ", Quantity:</pre>
 79
                 << itemQuantity << endl;
 80
        }
 81 };
 82
 83 - int main() {
        const int LIST_SIZE = 5; // Example size
 84
 85
        // Problem 2: Create an array GroceryList in the driver code
 86
        GroceryItem* groceryList[LIST_SIZE];
 87
         groceryList[0] = new Fruit("Apple", 1.50, 10);
 88
        groceryList[1] = new Vegetable("Carrot", 0.75, 5);
 89
         groceryList[2] = new Fruit("Banana", 1.20, 6);
 90
         groceryList[3] = new Vegetable("Lettuce", 2.00, 1);
 91
 92
         groceryList[4] = new Fruit("Orange", 1.80, 4);
 93
        // Display details about all items
 94
        cout << "Original Grocery List:" << endl;</pre>
 95
         for (int i = 0; i < LIST_SIZE; ++i) {
 96 +
            groceryList[i]->display();
 97
 98
 99
100
        double totalCost = 0.0;
101
102 ₹
         for (int i = 0; i < LIST_SIZE; ++i) {
             totalCost += groceryList[i]->calculateTotal();
103
104
        cout << "Total Cost: $" << totalCost << endl;</pre>
105
106
```

```
106
107
        // Problem 4: Delete the Lettuce from Jenna's GroceryList list and de-allocate th
            memory assigned.
108
        delete groceryList[3]; // Lettuce
109
        groceryList[3] = nullptr;
110
      cout << "Updated Grocery List:" << endl;</pre>
111
        for (int i = 0; i < LIST_SIZE; ++i) {
112 -
113 -
            if (groceryList[i] != nullptr) {
114
               groceryList[i]->display();
115
           }
116
        }
       for (int i = 0; i < LIST_SIZE; ++i) {
117 -
118 -
            if (groceryList[i] != nullptr) {
119
               delete groceryList[i];
120
           }
121
122
123 return 0;
124 }
125
```

Output

```
/tmp/4aUcZy6cuj.o
Original Grocery List:
Fruit: Apple, Price: $1.5, Quantity: 10
Vegetable: Carrot, Price: $0.75, Quantity: 5
Fruit: Banana, Price: $1.2, Quantity: 6
Vegetable: Lettuce, Price: $2, Quantity: 1
Fruit: Orange, Price: $1.8, Quantity: 4
Total Cost: $35.15
Updated Grocery List:
Fruit: Apple, Price: $1.5, Quantity: 10
Vegetable: Carrot, Price: $0.75, Quantity: 5
Fruit: Banana, Price: $1.2, Quantity: 6
Fruit: Orange, Price: $1.8, Quantity: 4
```

8. Conclusion

For this lab activity, I learned how to use class in C++. I was able to learn the constructors and deconstructors and their uses. Constructors are used for initializing values in an object. Destructors are used for deleting to save memory and prevent memory leakage. Further, I learned that arrays are also pointers. Arrays are like shortcuts for a group of pointers.

For the analysis of the procedure, In the foremost part of the program, the necessary header files were added. Below that is the class Student which will have the initialization of the variables and functions inside the class. After that are the constructors and destructors. In the main driver is the calling of the class and its initialized variables and functions.

For the supplementary activity, I did a similar procedure to the earlier activity. I used class, constructors, and destructors. However, I used a subclass under the main class to create an organized and understandable code. Under superclass is where I initialized the variables and functions needed. Then, I made the subclass inherit after the Superlass.

In this lab activity, I fell short on understanding C++ since I am only familiar with python when it comes to coding OOP. It was a hard activity for me since I had to self-study in a short time all the syntax and meaning of OOP in C++.

9. Assessment Rubric