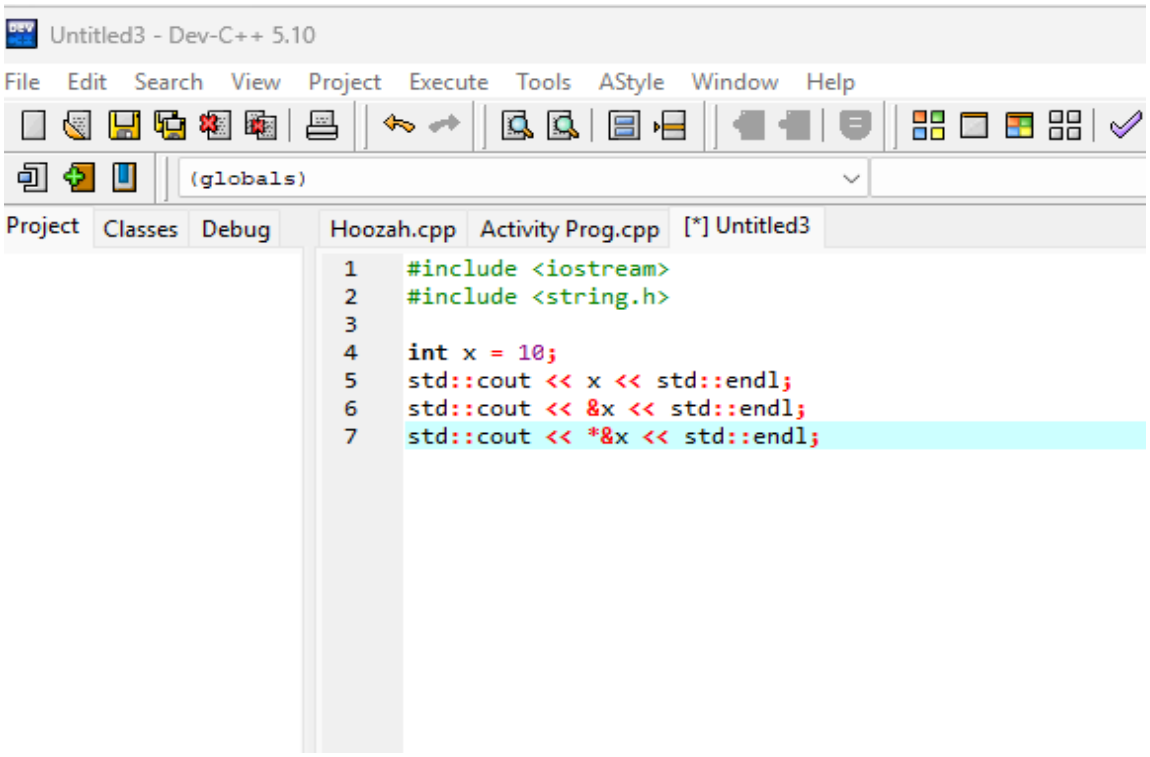
Activity No. <n>	
<Replace with Title>	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 7/31/2025**
Section: CPE21S4	**Date Submitted:**
Name(s): Anastacio, Lester Arvid P.	**Instructor: Jimlord Quejado**

6. Output

DISCUSSION



PROCEDURE

Hoozah.cpp

```cpp
#include <iostream>
#include <string.h>

class Student{
    private:
        std::string studentName;
        int studentAge;
    public:
    //constructor
    Student(std::string newName ="John Doe", int newAge=18){
        studentName = std::string(newName);
        studentAge = newAge;
        std::cout << "Constructor Called." << std::endl;
    };
        //deconstructor
    ~Student(){
        std::cout << "Destructor Called." << std::endl;
    }

    //Copy Constructor
    Student(const Student &copyStudent){
        std::cout << "Copy Constructor Called" << std::endl;
        studentName = copyStudent.studentName;
        studentAge = copyStudent.studentAge;
    }

    //Display Attributes
    void printDetails(){
        std::cout << this->studentName << " " << this->studentAge << std::endl;
    }

};
int main() {
    Student student1("Roman", 28);
    Student student2(student1);
    Student student3;
    student3 = student2;

    return 0;
}
```

```
C:\Users\TIPQC\Desktop\ANA
Constructor Called.
Copy Constructor Called
Constructor Called.
Destructor Called.
Destructor Called.
Destructor Called.

--------------------------------
Process exited after 0.014 seconds with return value 0
Press any key to continue . . .
```

Compile Log | Debug | Find Results | Close

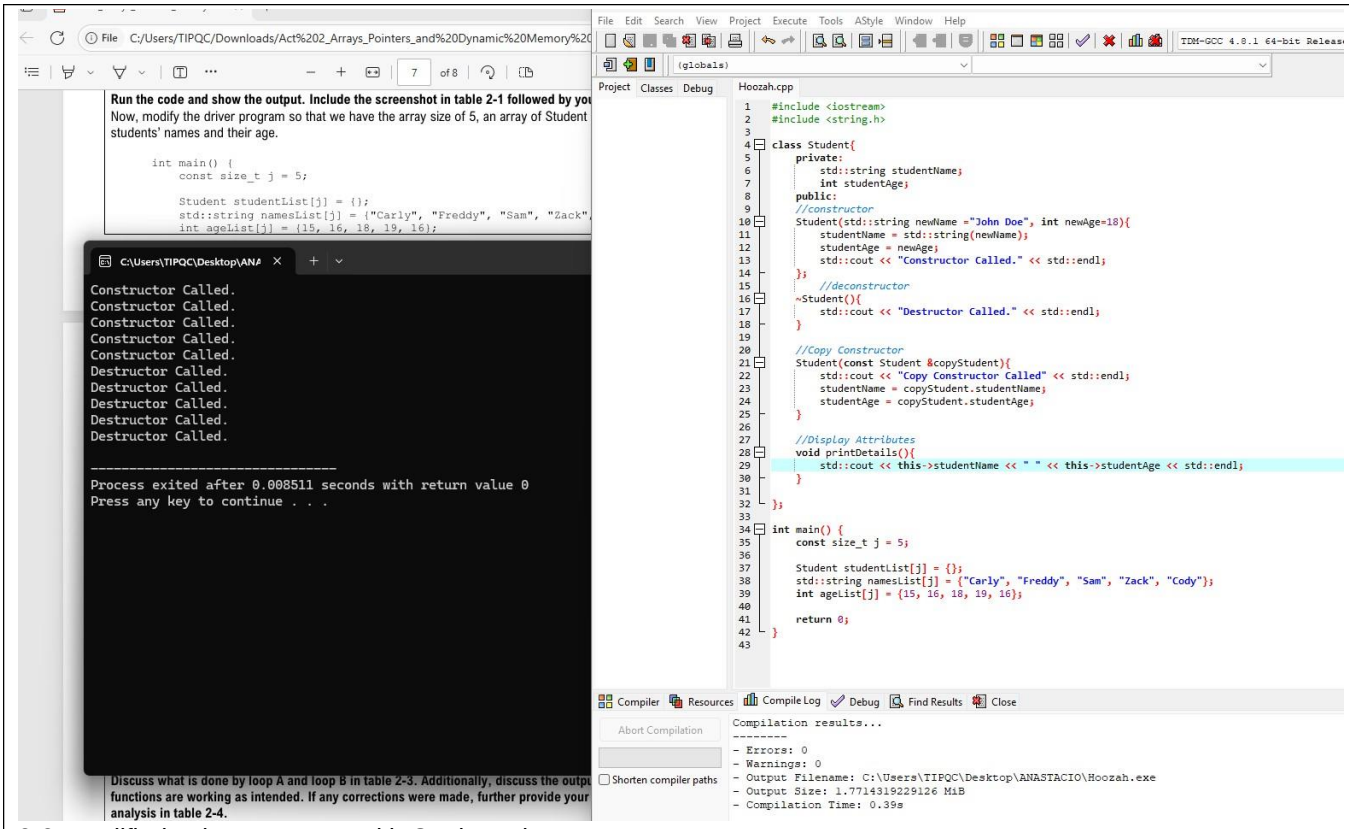
```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\TIPQC\Desktop\ANASTACIO\Hoozah.exe
- Output Size: 1.77234363555908 MiB
- Compilation Time: 0.39s
```

2.1.

Initial Driver Program

Observation:

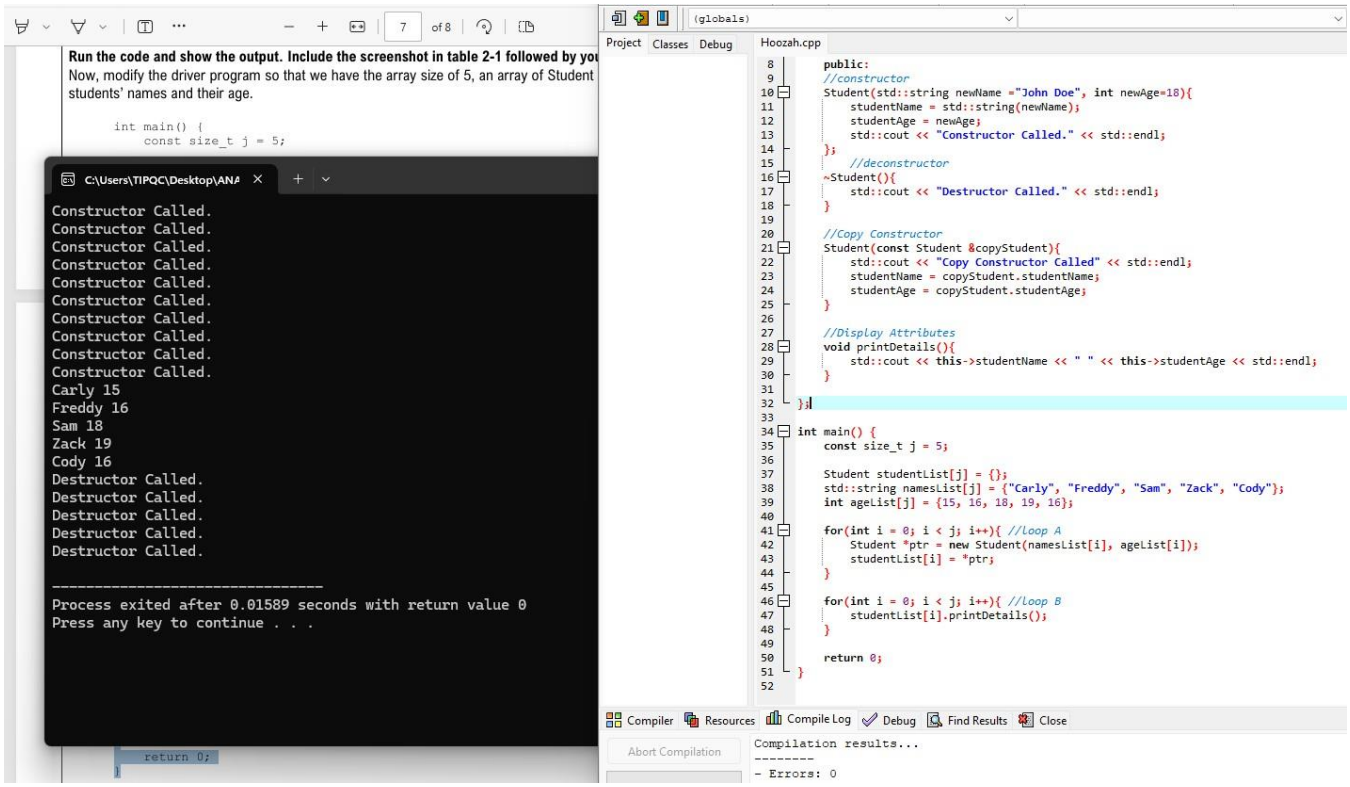
I've observed from this line of codes, shows how objects are being made and copied It uses a constructor, a copy constructor, and a destructor to show what happens when you create and delete objects.



2-2. Modified Driver Program with Student Lists

Observation:

I've observed that the amount of names listed inside the array, is the same amount of times that the constructor and destructor is being called, which in this case is 5 names, which resulted to both constructor and destructor being called the same amount.



2-3. Final Driver Program

Observation:

I've noticed that this time it finally showed the names and age of the data that is within the array, and the use of loops which is I am guessing used for a more organized output like what is shown in the screenshot.

BEFORE

```
public:
    //constructor
    Student(std::string newName ="John Doe", int newAge=18){
        studentName = std::move(newName);
        studentAge = newAge;
        std::cout << "Constructor Called." << std::endl;
    };
```

AFTER

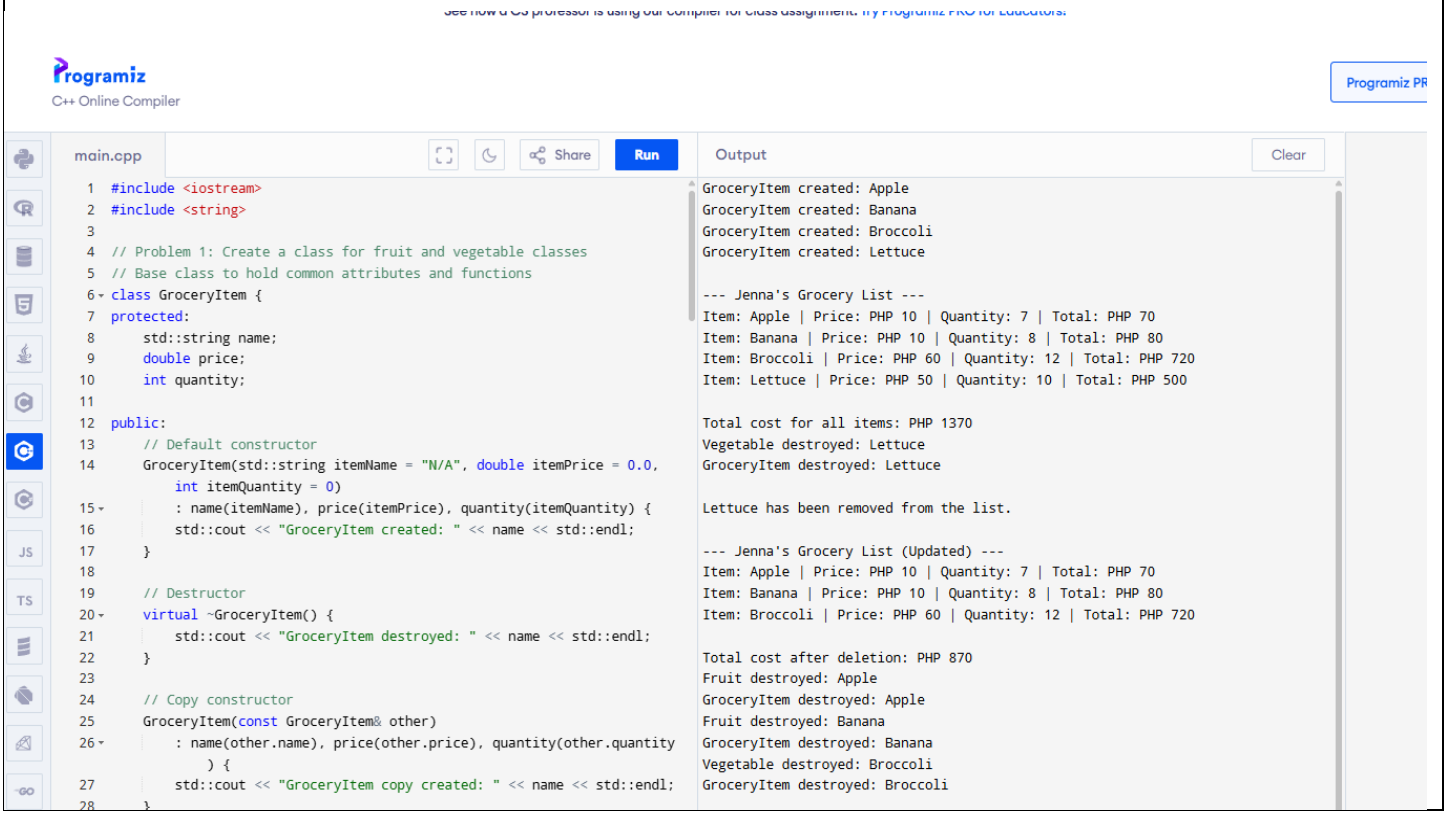
```
    int studentAge;
public:
//constructor
Student(std::string newName ="John Doe", int newAge=18){
    studentName = std::string(newName);
    studentAge = newAge;
    std::cout << "Constructor Called." << std::endl;
};
```

2-4. Modifications/Corrections Necessary

Summary:

I've made a tiny bit of a change here since the move function is stopping the code from working since it stated that move is not among the std, which is why I change it to string since it is amongst one of the functions added and just decided to give it a try since the names are string after all, and then now it works.

7. Supplementary Activity



```
29
30      // Copy assignment operator
31      GroceryItem& operator=(const GroceryItem& other) {
32          if (this != &other) {
33              name = other.name;
34              price = other.price;
35              quantity = other.quantity;
36          }
37          return *this;
38      }
39
40      // Function to calculate the sum for a single item
41      double calculateSum() const {
42          return price * quantity;
43      }
44
45      // Function to display item details
46      virtual void displayDetails() const {
47          std::cout << "Item: " << name << " | Price: PHP " << price
48                    << " | Quantity: " << quantity
49                    << " | Total: PHP " << calculateSum() << std::endl;
50      }
51
52      std::string getName() const { return name; }
53  };
54
55  // Derived Fruit class
56  class Fruit : public GroceryItem {
57  public:
58      Fruit(std::string itemName, double itemPrice, int itemQuantity)
```

```
GroceryItem created: Apple
GroceryItem created: Banana
GroceryItem created: Broccoli
GroceryItem created: Lettuce

--- Jenna's Grocery List ---
Item: Apple | Price: PHP 10 | Quantity: 7 | Total: PHP 70
Item: Banana | Price: PHP 10 | Quantity: 8 | Total: PHP 80
Item: Broccoli | Price: PHP 60 | Quantity: 12 | Total: PHP 720
Item: Lettuce | Price: PHP 50 | Quantity: 10 | Total: PHP 500

Total cost for all items: PHP 1370
Vegetable destroyed: Lettuce
GroceryItem destroyed: Lettuce

Lettuce has been removed from the list.

--- Jenna's Grocery List (Updated) ---
Item: Apple | Price: PHP 10 | Quantity: 7 | Total: PHP 70
Item: Banana | Price: PHP 10 | Quantity: 8 | Total: PHP 80
Item: Broccoli | Price: PHP 60 | Quantity: 12 | Total: PHP 720

Total cost after deletion: PHP 870
Fruit destroyed: Apple
GroceryItem destroyed: Apple
Fruit destroyed: Banana
GroceryItem destroyed: Banana
Vegetable destroyed: Broccoli
GroceryItem destroyed: Broccoli
```

main.cpp Share Run Output Clear

```
57  public:
58      Fruit(std::string itemName, double itemPrice, int itemQuantity)
59          : GroceryItem(itemName, itemPrice, itemQuantity) {}
60      ~Fruit() override {
61          std::cout << "Fruit destroyed: " << name << std::endl;
62      }
63  };
64
65  // Derived Vegetable class
66  class Vegetable : public GroceryItem {
67  public:
68      Vegetable(std::string itemName, double itemPrice, int itemQuantity)
69          : GroceryItem(itemName, itemPrice, itemQuantity) {}
70      ~Vegetable() override {
71          std::cout << "Vegetable destroyed: " << name << std::endl;
72      }
73  };
74
75  // Problem 3: Function to calculate total sum
76  double TotalSum(GroceryItem** list, int size) {
77      double total = 0.0;
78      for (int i = 0; i < size; ++i) {
79          if (list[i] != nullptr) {
80              total += list[i]->calculateSum();
81          }
82      }
83      return total;
84  }
85
```

```
GroceryItem created: Apple
GroceryItem created: Banana
GroceryItem created: Broccoli
GroceryItem created: Lettuce

--- Jenna's Grocery List ---
Item: Apple | Price: PHP 10 | Quantity: 7 | Total: PHP 70
Item: Banana | Price: PHP 10 | Quantity: 8 | Total: PHP 80
Item: Broccoli | Price: PHP 60 | Quantity: 12 | Total: PHP 720
Item: Lettuce | Price: PHP 50 | Quantity: 10 | Total: PHP 500

Total cost for all items: PHP 1370
Vegetable destroyed: Lettuce
GroceryItem destroyed: Lettuce

Lettuce has been removed from the list.

--- Jenna's Grocery List (Updated) ---
Item: Apple | Price: PHP 10 | Quantity: 7 | Total: PHP 70
Item: Banana | Price: PHP 10 | Quantity: 8 | Total: PHP 80
Item: Broccoli | Price: PHP 60 | Quantity: 12 | Total: PHP 720

Total cost after deletion: PHP 870
Fruit destroyed: Apple
GroceryItem destroyed: Apple
Fruit destroyed: Banana
GroceryItem destroyed: Banana
Vegetable destroyed: Broccoli
GroceryItem destroyed: Broccoli
```

main.cpp

Share | Run

```
}

int main() {
    const int listSize = 4;

    // Problem 2: Create an array GroceryList
    // Use an array of pointers to the base class for polymorphism
    GroceryItem** jennaGroceryList = new GroceryItem*[listSize];
    int currentSize = listSize;

    // Dynamically allocate Fruit and Vegetable objects
    jennaGroceryList[0] = new Fruit("Apple", 10, 7);
    jennaGroceryList[1] = new Fruit("Banana", 10, 8);
    jennaGroceryList[2] = new Vegetable("Broccoli", 60, 12);
    jennaGroceryList[3] = new Vegetable("Lettuce", 50, 10);

    std::cout << "\n--- Jenna's Grocery List ---" << std::endl;
    for (int i = 0; i < currentSize; ++i) {
        jennaGroceryList[i]->displayDetails();
    }

    double totalCost = TotalSum(jennaGroceryList, currentSize);
    std::cout << "\nTotal cost for all items: PHP " << totalCost << std
        ::endl;

    // Problem 4: Delete Lettuce and de-allocate memory
    std::string itemToDelete = "Lettuce";
    int indexToDelete = -1;
    for (int i = 0; i < currentSize; ++i) {
        if (jennaGroceryList[i]->getName() == itemToDelete) {
```

Output | Clear

```
GroceryItem created: Apple
GroceryItem created: Banana
GroceryItem created: Broccoli
GroceryItem created: Lettuce

--- Jenna's Grocery List ---
Item: Apple | Price: PHP 10 | Quantity: 7 | Total: PHP 70
Item: Banana | Price: PHP 10 | Quantity: 8 | Total: PHP 80
Item: Broccoli | Price: PHP 60 | Quantity: 12 | Total: PHP 720
Item: Lettuce | Price: PHP 50 | Quantity: 10 | Total: PHP 500

Total cost for all items: PHP 1370
Vegetable destroyed: Lettuce
GroceryItem destroyed: Lettuce

Lettuce has been removed from the list.

--- Jenna's Grocery List (Updated) ---
Item: Apple | Price: PHP 10 | Quantity: 7 | Total: PHP 70
Item: Banana | Price: PHP 10 | Quantity: 8 | Total: PHP 80
Item: Broccoli | Price: PHP 60 | Quantity: 12 | Total: PHP 720

Total cost after deletion: PHP 870
Fruit destroyed: Apple
GroceryItem destroyed: Apple
Fruit destroyed: Banana
GroceryItem destroyed: Banana
Vegetable destroyed: Broccoli
GroceryItem destroyed: Broccoli
```

main.cpp

Share | Run

```
    double totalCost = TotalSum(jennaGroceryList, currentSize);
    std::cout << "\nTotal cost for all items: PHP " << totalCost << std
        ::endl;

    // Problem 4: Delete Lettuce and de-allocate memory
    std::string itemToDelete = "Lettuce";
    int indexToDelete = -1;
    for (int i = 0; i < currentSize; ++i) {
        if (jennaGroceryList[i]->getName() == itemToDelete) {
            indexToDelete = i;
            break;
        }
    }

    if (indexToDelete != -1) {
        delete jennaGroceryList[indexToDelete];
        std::cout << "\n" << itemToDelete << " has been removed from
            the list." << std::endl;
        // Shift remaining elements to the left to fill the gap
        for (int i = indexToDelete; i < currentSize - 1; ++i) {
            jennaGroceryList[i] = jennaGroceryList[i + 1];
        }
        jennaGroceryList[currentSize - 1] = nullptr;
        currentSize--;
    } else {
        std::cout << "\nItem to delete not found." << std::endl;
    }

    std::cout << "\n--- Jenna's Grocery List (Updated) ---" << std
        ::endl;
```

Output | Clear

```
GroceryItem created: Apple
GroceryItem created: Banana
GroceryItem created: Broccoli
GroceryItem created: Lettuce

--- Jenna's Grocery List ---
Item: Apple | Price: PHP 10 | Quantity: 7 | Total: PHP 70
Item: Banana | Price: PHP 10 | Quantity: 8 | Total: PHP 80
Item: Broccoli | Price: PHP 60 | Quantity: 12 | Total: PHP 720
Item: Lettuce | Price: PHP 50 | Quantity: 10 | Total: PHP 500

Total cost for all items: PHP 1370
Vegetable destroyed: Lettuce
GroceryItem destroyed: Lettuce

Lettuce has been removed from the list.

--- Jenna's Grocery List (Updated) ---
Item: Apple | Price: PHP 10 | Quantity: 7 | Total: PHP 70
Item: Banana | Price: PHP 10 | Quantity: 8 | Total: PHP 80
Item: Broccoli | Price: PHP 60 | Quantity: 12 | Total: PHP 720

Total cost after deletion: PHP 870
Fruit destroyed: Apple
GroceryItem destroyed: Apple
Fruit destroyed: Banana
GroceryItem destroyed: Banana
Vegetable destroyed: Broccoli
GroceryItem destroyed: Broccoli
```

main.cpp

Share | Run

```
            the list." << std::endl;
        // Shift remaining elements to the left to fill the gap
        for (int i = indexToDelete; i < currentSize - 1; ++i) {
            jennaGroceryList[i] = jennaGroceryList[i + 1];
        }
        jennaGroceryList[currentSize - 1] = nullptr;
        currentSize--;
    } else {
        std::cout << "\nItem to delete not found." << std::endl;
    }

    std::cout << "\n--- Jenna's Grocery List (Updated) ---" << std
        ::endl;
    for (int i = 0; i < currentSize; ++i) {
        if (jennaGroceryList[i] != nullptr) {
            jennaGroceryList[i]->displayDetails();
        }
    }

    double updatedTotalCost = TotalSum(jennaGroceryList, currentSize);
    std::cout << "\nTotal cost after deletion: PHP " <<
        updatedTotalCost << std::endl;

    // De-allocate the remaining memory
    for (int i = 0; i < currentSize; ++i) {
        delete jennaGroceryList[i];
    }
    delete[] jennaGroceryList;
```

Output | Clear

```
GroceryItem created: Apple
GroceryItem created: Banana
GroceryItem created: Broccoli
GroceryItem created: Lettuce

--- Jenna's Grocery List ---
Item: Apple | Price: PHP 10 | Quantity: 7 | Total: PHP 70
Item: Banana | Price: PHP 10 | Quantity: 8 | Total: PHP 80
Item: Broccoli | Price: PHP 60 | Quantity: 12 | Total: PHP 720
Item: Lettuce | Price: PHP 50 | Quantity: 10 | Total: PHP 500

Total cost for all items: PHP 1370
Vegetable destroyed: Lettuce
GroceryItem destroyed: Lettuce

Lettuce has been removed from the list.

--- Jenna's Grocery List (Updated) ---
Item: Apple | Price: PHP 10 | Quantity: 7 | Total: PHP 70
Item: Banana | Price: PHP 10 | Quantity: 8 | Total: PHP 80
Item: Broccoli | Price: PHP 60 | Quantity: 12 | Total: PHP 720

Total cost after deletion: PHP 870
Fruit destroyed: Apple
GroceryItem destroyed: Apple
Fruit destroyed: Banana
GroceryItem destroyed: Banana
Vegetable destroyed: Broccoli
GroceryItem destroyed: Broccoli
```

8. Conclusion

I did pretty well during this activity, there is much more to improve though since I struggled a lot, especially since this took me longer than I intended since I once quite proficient with c++ during the 1st year, but I did learn a lot though, I learned more about constructors, destructors, copy and etc., during the activity too, I used different platforms and wikis and information so that I can finally finish this task.

9. Assessment Rubric