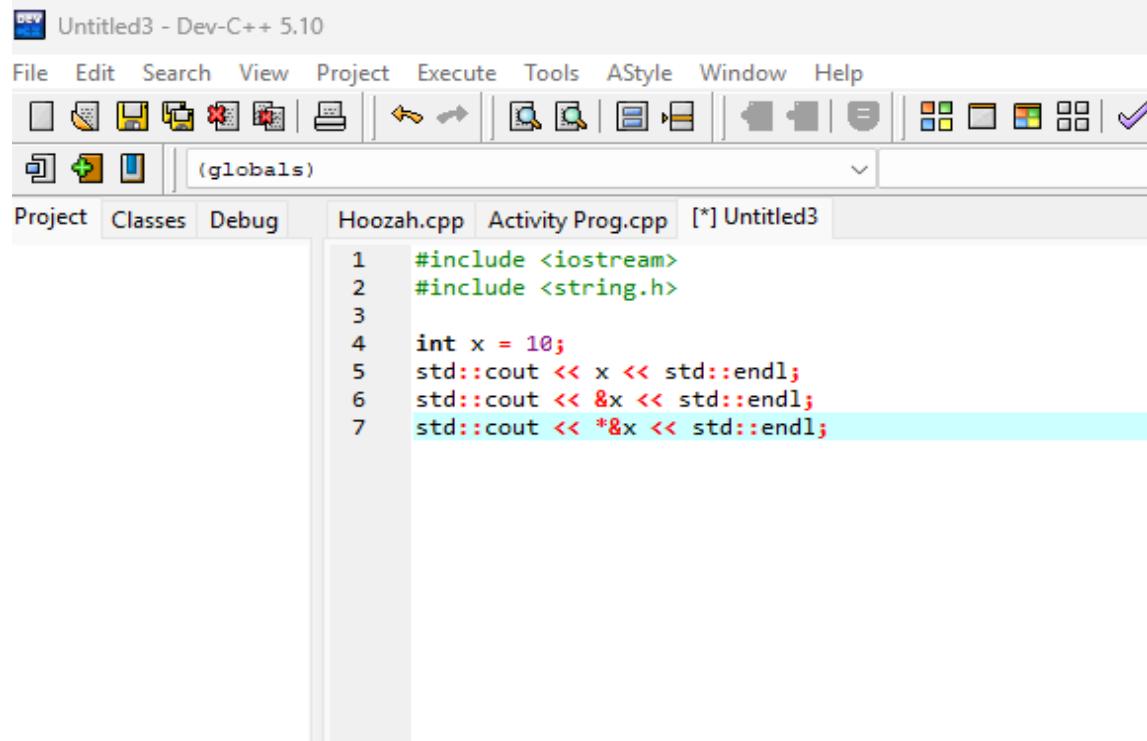


Activity No. <n>**<Replace with Title>**

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**

The screenshot shows the Dev-C++ 5.10 IDE interface. The title bar reads "Untitled3 - Dev-C++ 5.10". The menu bar includes File, Edit, Search, View, Project, Execute, Tools, AStyle, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Print, and Build. Below the toolbar is a status bar showing "(globals)". The tab bar at the bottom has tabs for "Project", "Classes", "Debug", "Hoozah.cpp", "Activity Prog.cpp", and "[*] Untitled3". The main code editor window displays the following C++ code:

```
1 #include <iostream>
2 #include <string.h>
3
4 int x = 10;
5 std::cout << x << std::endl;
6 std::cout << &x << std::endl;
7 std::cout << *&x << std::endl;
```

PROCEDURE

The screenshot shows a C++ development environment with two windows. The left window displays the source code for `Hoozah.cpp`. The right window shows the terminal output of the program execution.

Code (Hoozah.cpp):

```
1 #include <iostream>
2 #include <string.h>
3
4 class Student{
5 private:
6     std::string studentName;
7     int studentAge;
8 public:
9     //constructor
10    Student(std::string newName = "John Doe", int newAge=18){
11        studentName = std::string(newName);
12        studentAge = newAge;
13        std::cout << "Constructor Called." << std::endl;
14    }
15    //deconstructor
16    ~Student(){
17        std::cout << "Destructor Called." << std::endl;
18    }
19
20    //Copy Constructor
21    Student(const Student &copyStudent){
22        std::cout << "Copy Constructor Called" << std::endl;
23        studentName = copyStudent.studentName;
24        studentAge = copyStudent.studentAge;
25    }
26
27    //Display Attributes
28    void printDetails(){
29        std::cout << this->studentName << " " << this->studentAge << std::endl;
30    }
31 }
32
33
34 int main(){
35     Student student1("Roman", 28);
36     Student student2(student1);
37     Student student3;
38     student3 = student2;
39
40     return 0;
41 }
```

Terminal Output:

```
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 . . .
```

Compilation Results:

```
Compilation results...
-----
- Errors: 0
- Warnings: 0
ths - Output Filename: C:\Users\TIPQC\Desktop\ANASTACIO\Hoozah.exe
- Output Size: 1.7723436555908 MiB
- Compilation Time: 0.39s
```

Section-Header:

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.

```

Run the code and show the output. Include the screenshot in table 2-1 followed by you.
Now, modify the driver program so that we have the array size of 5, an array of Student
students' names and their age.

int main() {
    const size_t j = 5;

    Student studentList[j] = {};
    std::string namesList[j] = {"Carly", "Freddy", "Sam", "Zack"};
    int ageList[j] = {15, 16, 18, 19, 16};

Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.

Process exited after 0.008511 seconds with return value 0
Press any key to continue . .

```

```

1 #include <iostream>
2 #include <string.h>
3
4 class Student{
5     private:
6         std::string studentName;
7         int studentAge;
8     public:
9         //constructor
10        Student(std::string newName = "John Doe", int newAge=18){
11            studentName = std::string(newName);
12            studentAge = newAge;
13            std::cout << "Constructor Called." << std::endl;
14        }
15        //destructor
16        ~Student(){
17            std::cout << "Destructor Called." << std::endl;
18        }
19
20        //Copy Constructor
21        Student(const Student &copyStudent){
22            std::cout << "Copy Constructor Called" << std::endl;
23            studentName = copyStudent.studentName;
24            studentAge = copyStudent.studentAge;
25        }
26
27        //Display Attributes
28        void printDetails(){
29            std::cout << this->studentName << " " << this->studentAge << std::endl;
30        }
31
32    };
33
34 int main() {
35     const size_t j = 5;

36     Student studentList[j] = {};
37     std::string namesList[j] = {"Carly", "Freddy", "Sam", "Zack", "Cody"};
38     int ageList[j] = {15, 16, 18, 19, 16};
39
40     return 0;
41 }
42

```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\TIPQC\Desktop\ANASTACIO\Hoozah.exe
- Output Size: 1.771431829126 MiB
- Compilation Time: 0.39s

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.

```

Run the code and show the output. Include the screenshot in table 2-1 followed by you.
Now, modify the driver program so that we have the array size of 5, an array of Student
students' names and their age.

int main() {
    const size_t j = 5;

Constructor Called.
Carly 15
Freddy 16
Sam 18
Zack 19
Cody 16
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.

Process exited after 0.01589 seconds with return value 0
Press any key to continue . .

```

```

8
9     public:
10    //constructor
11    Student(std::string newName = "John Doe", int newAge=18){
12        studentName = std::string(newName);
13        studentAge = newAge;
14        std::cout << "Constructor Called." << std::endl;
15    }
16    //destructor
17    ~Student(){
18        std::cout << "Destructor Called." << std::endl;
19    }
20
21    //Copy Constructor
22    Student(const Student &copyStudent){
23        std::cout << "Copy Constructor Called" << std::endl;
24        studentName = copyStudent.studentName;
25        studentAge = copyStudent.studentAge;
26    }
27
28    //Display Attributes
29    void printDetails(){
30        std::cout << this->studentName << " " << this->studentAge << std::endl;
31    }
32};

33
34 int main() {
35     const size_t j = 5;

36     Student studentList[j] = {};
37     std::string namesList[j] = {"Carly", "Freddy", "Sam", "Zack", "Cody"};
38     int ageList[j] = {15, 16, 18, 19, 16};

41     for(int i = 0; i < j; i++){ //Loop A
42         Student *ptr = new Student(namesList[i], ageList[i]);
43         studentList[i] = *ptr;
44     }

46     for(int i = 0; i < j; i++){ //Loop B
47         studentList[i].printDetails();
48     }

49
50     return 0;
51 }

return 0;

```

Compilation results...

- Errors: 0

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

See how a C++ processor is using our compiler for class assignments. Try PROGRAMIZ PRO FOR EDUCATORS!

Programiz
C++ Online Compiler

Programiz PR

```
main.cpp
```

```
1 #include <iostream>
2 #include <string>
3
4 // Problem 1: Create a class for fruit and vegetable classes
5 // Base class to hold common attributes and functions
6 class GroceryItem {
7 protected:
8     std::string name;
9     double price;
10    int quantity;
11
12 public:
13     // Default constructor
14     GroceryItem(std::string itemName = "N/A", double itemPrice = 0.0,
15         int itemQuantity = 0)
16         : name(itemName), price(itemPrice), quantity(itemQuantity)
17     {
18         std::cout << "GroceryItem created: " << name << std::endl;
19     }
20
21     // Destructor
22     virtual ~GroceryItem() {
23         std::cout << "GroceryItem destroyed: " << name << std::endl;
24     }
25
26     // Copy constructor
27     GroceryItem(const GroceryItem& other)
28         : name(other.name), price(other.price), quantity(other.quantity)
29     {
30         std::cout << "GroceryItem copy created: " << name << std::endl;
31     }
32 }
```

Run

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
```

```

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
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

Run Output Clear

```

84 }
85
86 int main() {
87     const int listSize = 4;
88
89     // Problem 2: Create an array GroceryList
90     // Use an array of pointers to the base class for polymorphism
91     GroceryItem** jennaGroceryList = new GroceryItem*[listSize];
92     int currentSize = listSize;
93
94     // Dynamically allocate Fruit and Vegetable objects
95     jennaGroceryList[0] = new Fruit("Apple", 10, 7);
96     jennaGroceryList[1] = new Fruit("Banana", 10, 8);
97     jennaGroceryList[2] = new Vegetable("Broccoli", 60, 12);
98     jennaGroceryList[3] = new Vegetable("Lettuce", 50, 10);
99
100    std::cout << "\n--- Jenna's Grocery List ---" << std::endl;
101    for (int i = 0; i < currentSize; ++i) {
102        jennaGroceryList[i]->displayDetails();
103    }
104
105    double totalCost = TotalSum(jennaGroceryList, currentSize);
106    std::cout << "\nTotal cost for all items: PHP " << totalCost << std
107 :endl;
108
109     // Problem 4: Delete Lettuce and de-allocate memory
110     std::string itemToDelete = "Lettuce";
111     int indexToDelete = -1;
112     for (int i = 0; i < currentSize; ++i) {
113         if (jennaGroceryList[i]->getName() == itemToDelete) {
114             indexToDelete = i;
115             break;
116         }
117
118         if (indexToDelete != -1) {
119             delete jennaGroceryList[indexToDelete];
120             std::cout << "\n" << itemToDelete << " has been removed from
121             the list." << std::endl;
122             // Shift remaining elements to the left to fill the gap
123             for (int i = indexToDelete; i < currentSize - 1; ++i) {
124                 jennaGroceryList[i] = jennaGroceryList[i + 1];
125             }
126             jennaGroceryList[currentSize - 1] = nullptr;
127             currentSize--;
128         } else {
129             std::cout << "\nItem to delete not found." << std::endl;
130         }
131     std::cout << "\n--- Jenna's Grocery List (Updated) ---" << std
132 :endl;

```

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

Run Output Clear

```

105     double totalCost = TotalSum(jennaGroceryList, currentSize);
106     std::cout << "\nTotal cost for all items: PHP " << totalCost << std
107 :endl;
108
109     // Problem 4: Delete Lettuce and de-allocate memory
110     std::string itemToDelete = "Lettuce";
111     int indexToDelete = -1;
112     for (int i = 0; i < currentSize; ++i) {
113         if (jennaGroceryList[i]->getName() == itemToDelete) {
114             indexToDelete = i;
115             break;
116         }
117
118         if (indexToDelete != -1) {
119             delete jennaGroceryList[indexToDelete];
120             std::cout << "\n" << itemToDelete << " has been removed from
121             the list." << std::endl;
122             // Shift remaining elements to the left to fill the gap
123             for (int i = indexToDelete; i < currentSize - 1; ++i) {
124                 jennaGroceryList[i] = jennaGroceryList[i + 1];
125             }
126             jennaGroceryList[currentSize - 1] = nullptr;
127             currentSize--;
128         } else {
129             std::cout << "\nItem to delete not found." << std::endl;
130         }
131     std::cout << "\n--- Jenna's Grocery List (Updated) ---" << std
132 :endl;

```

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

    the list." << std::endl;
121 // Shift remaining elements to the left to fill the gap
122+ for (int i = indexToDelete; i < currentSize - 1; ++i) {
123     jennaGroceryList[i] = jennaGroceryList[i + 1];
124 }
125 jennaGroceryList[currentSize - 1] = nullptr;
126 currentSize--;
127+ } else {
128     std::cout << "\nItem to delete not found." << std::endl;
129 }
130
131 std::cout << "\n--- Jenna's Grocery List (Updated) ---" << std::endl;
132+ for (int i = 0; i < currentSize; ++i) {
133+     if (jennaGroceryList[i] != nullptr) {
134         jennaGroceryList[i]->displayDetails();
135     }
136 }
137
138 double updatedTotalCost = TotalSum(jennaGroceryList, currentSize);
139 std::cout << "\nTotal cost after deletion: PHP " <<
140     updatedTotalCost << std::endl;
141
142+ // De-allocate the remaining memory
143+ for (int i = 0; i < currentSize; ++i) {
144     delete jennaGroceryList[i];
145 }
146 delete[] jennaGroceryList;
147

```

Output

```

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