Activity No. 2.1	
Arrays, Pointers and Dynamic Memory Allocation	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 9/11/24**
Section: CPE21S1	**Date Submitted: 9/11/24**
Name(s): Kurt Gabriel Anduque	**Instructor: Mrs. Maria Rizette Sayo**

6. Output

whole code:

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
#include <string>
using namespace std;


    class Student{
        private:
        string studentName;
        int studentAge;

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

        //deconstructor
        ~Student(){
            cout << "Destructor Called." << endl;}
        //Copy Constructor
        Student(const Student &copyStudent){
        cout << "Copy Constructor Called" << endl;
        studentName = copyStudent.studentName;
        studentAge = copyStudent.studentAge;
}
        //Display Attributes
         void printDetails(){
        cout << this->studentName << " " << this->studentAge << endl;
}
};


int main() {
const size_t j = 5;
Student studentList[j] = {};
std::string namesList[j] = {"Carly", "Freddy", "Sam", "Zack", "Cody"};
int ageList[j] = {15, 16, 18, 19, 16};
for(int i = 0; i < j; i++){ //loop A
Student *ptr = new Student(namesList[i], ageList[i]);
studentList[i] = *ptr;
}
for(int i = 0; i < j; i++){ //loop B
studentList[i].printDetails();
}
return 0;
```

Table 2.1 Initial Drivers

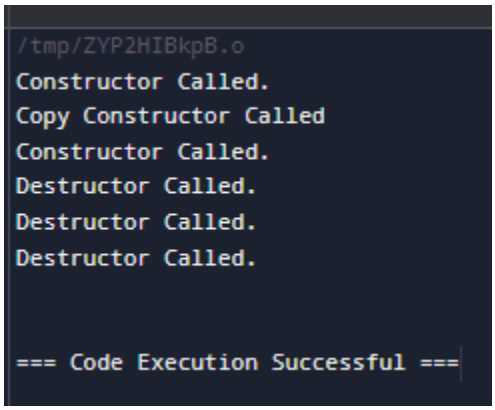
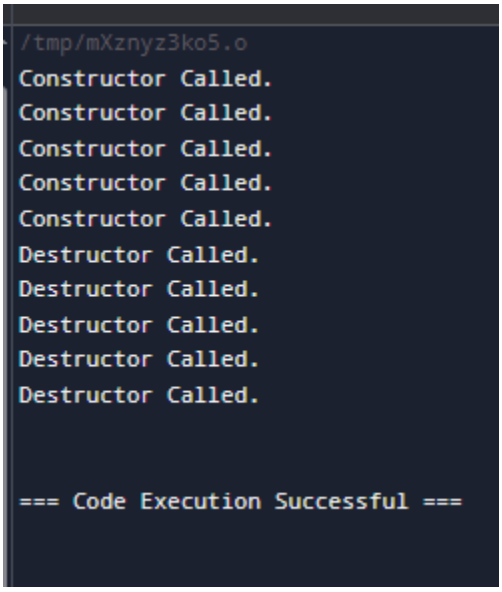
Screenshot	
Observation	**Based on my observation the constructor was called**

Table 2-2. Modified Driver Program with Student Lists

Screenshot	
Observation	**Because we used numerous instances of the array, the constructor and destructor are successfully called numerous times in this output.**

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

Observation	**The pointer is used in the first loop to maintain the allocated memory for the array.**

Loob B	
	`for(int i = 0; i < j; i++){`
	`  studentList[i].printDetails();`
	`}`

Observation	**The loop B accesses the memory that was previously allocated.**

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

Observation	**Now that the arrays have been allocated statically, we may access and dynamically save them using the class.**

Table 2-3. Final Driver Program

7. Supplementary Activity

main.cpp

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

using namespace std;

class Fruit {
public:

    Fruit(string name = "", double price = 0.0, int quantity = 0)
        : name_(name), price_(price), quantity_(quantity) {}


    ~Fruit() {
        cout << "Fruit destructor called" << endl;
    }


    Fruit(const Fruit& other)
        : name_(other.name_), price_(other.price_), quantity_(other.quantity_) {
        cout << "Fruit copy constructor called" << endl;
    }


    Fruit& operator=(const Fruit& other) {
        if (this != &other) {
            name_ = other.name_;
            price_ = other.price_;
            quantity_ = other.quantity_;
        }
        cout << "Fruit copy assignment operator called" << endl;
        return *this;
    }


    string name_;
    double price_;
```

Output

```
/tmp/em9vz2TrVO.o
Grocery List:
Fruit: Apple, Price: 10, Quantity: 7
Fruit: Banana, Price: 10, Quantity: 8
Vegetable: Broccoli, Price: 60, Quantity: 12
Vegetable: Lettuce, Price: 50, Quantity: 10
Total cost: 1370
Lettuce removed from the list.

Grocery List after removal:
Fruit: Apple, Price: 10, Quantity: 7
Fruit: Banana, Price: 10, Quantity: 8
Vegetable: Broccoli, Price: 60, Quantity: 12
Vegetable destructor called
Vegetable destructor called
Fruit destructor called
Fruit destructor called


=== Code Execution Successful ===
```

8. Conclusion

- **I gained knowledge about C++ class and constructor creation from this exercise. They are excellent for managing data. I gained knowledge about handling both dynamic and static data storage in arrays. In order to access the memory that an object has allocated, pointers are also used. The task guides you through the process of creating the software using dynamic memory step-by-step. I discovered that constructors are useful for quickly adding properties and creating objects, while deconstructors are useful for identifying objects in memory.**

9. Assessment Rubric