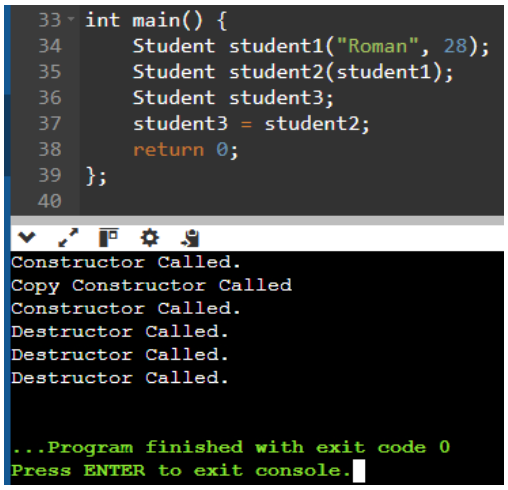
Activity No. 2	
Arrays, Pointers and Dynamic Memory Allocation	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** 09 / 11 / 2024
Section: CPE21S4	**Date Submitted:** 09 / 13 / 2024
Name(s): ROALLOS, Jean Gabriel Vincent G.	**Instructor:** Prof. Maria Rizette M. Sayo

6. Output

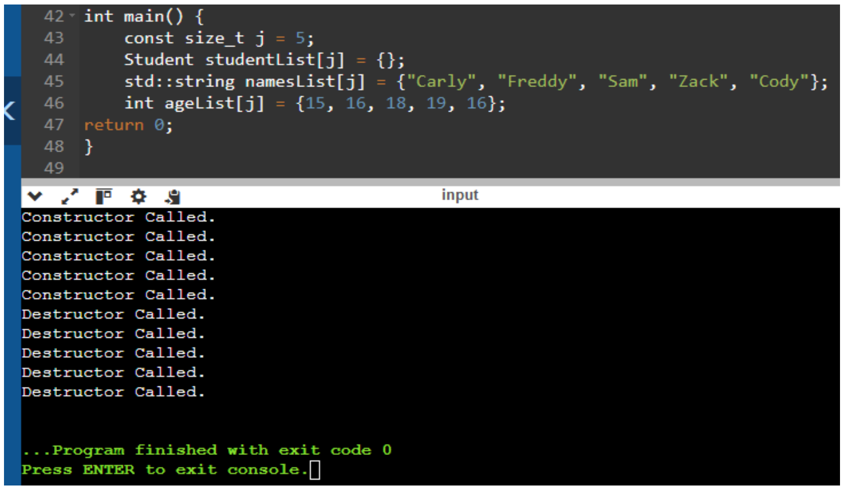
Screenshot	
Observation	The code successfully ran and the necessary status prints were correct.

```cpp
int main() {
    Student student1("Roman", 28);
    Student student2(student1);
    Student student3;
    student3 = student2;
    return 0;
};
```

```
Constructor Called.
Copy Constructor Called
Constructor Called.
Destructor Called.
Destructor Called.
Destructor Called.


...Program finished with exit code 0
Press ENTER to exit console.
```

Table 2-1. Initial Driver Program

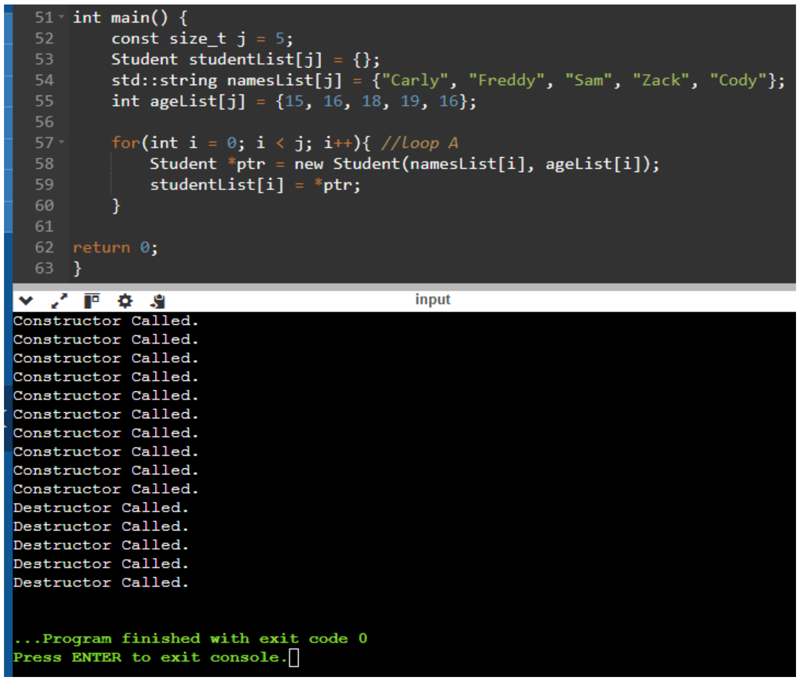
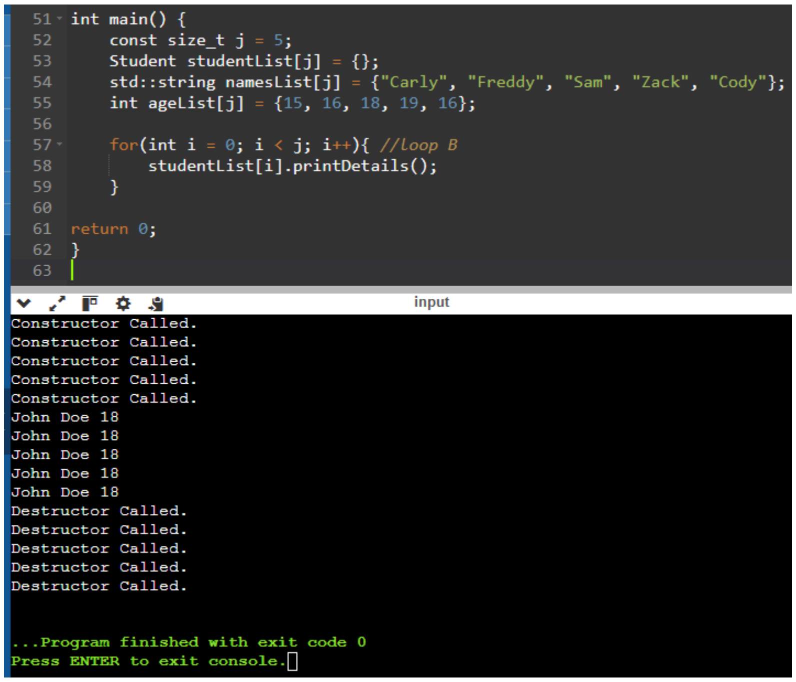
Screenshot	
Observation	Both the constructor and the deconstructor processes occurred and it is the same as the number of items in the array.

```cpp
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};
return 0;
}
```

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


...Program finished with exit code 0
Press ENTER to exit console.
```

Table 2-2. Modified Driver Program with Student Lists

Loop A	
Observation	The constructor instances were doubled and the destructor instances count remained the same.
Loop B	
Observation	The information stored in the public section of the class “Student” was constructed, printed, and deconstructed 5 times, following the defined size.

```cpp
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;
    }

return 0;
}
```

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


...Program finished with exit code 0
Press ENTER to exit console.
```

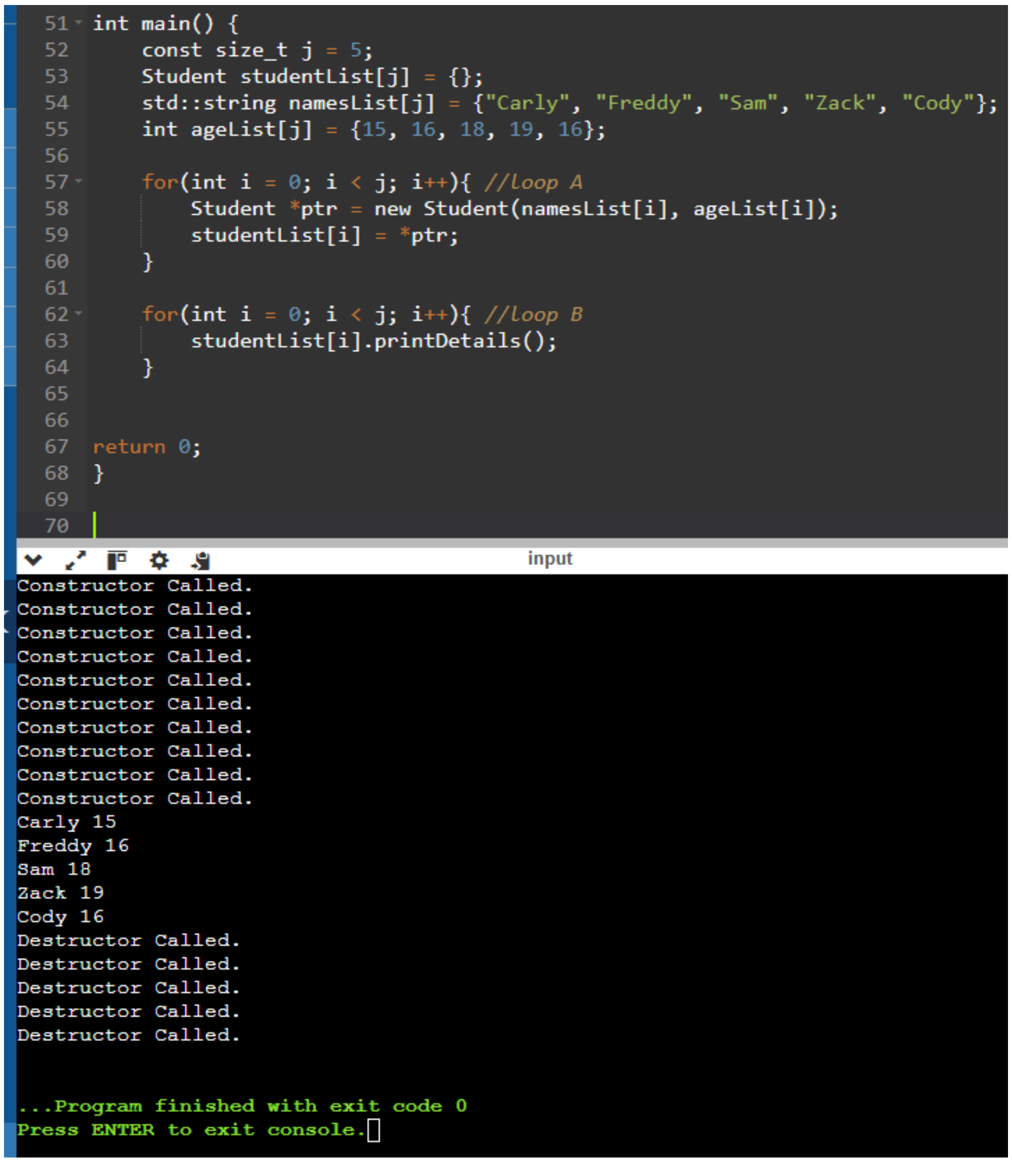
```cpp
int main() {
    const size_t j = 5;
    Student studentList[j] = {};
    std::string namesList[j] = {"Carly", "Freddy", "Sam", "Zack", "Cody"};
    int ageList[j] = {15, 16, 18, 19, 16};

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

return 0;
}
```

```
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
John Doe 18
John Doe 18
John Doe 18
John Doe 18
John Doe 18
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.


...Program finished with exit code 0
Press ENTER to exit console.
```

Output	

```
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;
}

```

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


...Program finished with exit code 0
Press ENTER to exit console.
```

Observation	It is a combination of both loops A and B, in which there are 10 constructor instances that happened, print the combined contents of the namesList and ageList, and 5 deconstructor instances.

Table 2-3. Final Driver Program

Modification	N / A
Observation	There were no modifications done.

Table 2-4. Modifications/Corrections Necessary

7. Supplementary Activity

```
#include <iostream>
#include <string.h>
using namespace std;

class Fruit {

    private:
        string fruitName;
        double fruitPrice;
        int fruitQuantity;

    public:
        //constructor
        Fruit (string name, double price, int quantity) {
            fruitName = move(name);
            fruitPrice = price;
            fruitQuantity = quantity;
            cout << "Fruit Constructor called." << endl;
        }

        //deconstructor
        ~Fruit(){
            cout << "Fruit Deconstructor called." << endl;
        }

        //copy constructor
        Fruit (const Fruit &copyFruit) {
            cout << "Fruit Copy Constructor called." << endl;
            fruitName = copyFruit.fruitName;
            fruitPrice = copyFruit.fruitPrice;
            fruitQuantity = copyFruit.fruitQuantity;
        }

        double sumCalc() {
            return fruitPrice * fruitQuantity;
        }

        void displayInfo() {
            cout << "Fruit: " << fruitName << ", Price: " << fruitPrice << ", Quantity: " <<
fruitQuantity << endl;
        }
};

class Vegetable {

    private:
        string veggieName;
        double veggiePrice;
        int veggieQuantity;

    public:
        //constructor
        Vegetable (string name, double price, int quantity) {
            veggieName = move(name);
            veggiePrice = price;
            veggieQuantity = quantity;
            cout << "Vegetable Constructor called." << endl;
        }

        //deconstructor
        ~Vegetable() {
            cout << "Vegetable Deconstructor called." << endl;
        }

        //copy constructor
        Vegetable (const Vegetable &copyVeggie) {
            cout << "Vegetable Copy Constructor called." << endl;
            veggieName = copyVeggie.veggieName;
            veggiePrice = copyVeggie.veggiePrice;
            veggieQuantity = copyVeggie.veggieQuantity;
        }

        double sumCalc() {
            return veggiePrice * veggieQuantity;
        }

        void displayInfo() {
            cout << "Vegetable: " << veggieName << ", Price: " << veggiePrice << ", Quantity: " <<
veggieQuantity << endl;
```

```
        }

};

int main() {

    //initializations
    Fruit apple ("Apple", 10, 7);
    apple.displayInfo();
    cout << "Total cost: " << apple.sumCalc() << endl;

    Fruit banana ("Banana", 10, 8);
    banana.displayInfo();
    cout << "Total cost: " << banana.sumCalc() << endl;

    Vegetable brocolli ("Brocolli", 60, 12);
    brocolli.displayInfo();
    cout << "Total cost: " << brocolli.sumCalc() << endl;

    Vegetable lettuce ("Lettuce", 50, 10);
    lettuce.displayInfo();
    cout << "Total cost: " << lettuce.sumCalc() << endl;

    double totalSum = apple.sumCalc() + banana.sumCalc() + brocolli.sumCalc() + lettuce.sumCalc();

    cout << "Total List cost: " << totalSum << endl;

    return 0;
}
```

8. Conclusion

I have learned about using classes, pointers and different operations used between them. I had quite a hard time understanding a few things about the syntaxes I was not familiar with within C++, but I think I have managed to grasp it. Although the supplementary activity is fairly done, I think there are things to optimize, such as utilizing class inheritances, so the code would not be as long and repetitive.

9. Assessment Rubric