

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

<b>Course Code:</b> CPE010	<b>Program:</b> Computer Engineering
<b>Course Title:</b> Data Structures and Algorithms	<b>Date Performed:</b> 08/06/25
<b>Section:</b> CPE 010-CPE21S4	<b>Date Submitted:</b> 08/06/25
<b>Name(s):</b> Catungal Kerwin Jan B	<b>Instructor:</b> Jimlord Quejado

**6. Output****Discussion:**

1.cpp Discussion.cpp Discussion array.cpp

```
1 #include <iostream>
2 #include <string>
3
4 int main(){
5     int x = 10;
6     std::cout << x << std::endl;
7     std::cout << &x << std::endl;
8     std::cout << *x << std::endl;
9
10 }
11
```

:pp Discussion.cpp Discussion array.cpp

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

int main(){

    int array[] = {1, 2, 3, 4};
    int *ptrArray
    //array = ptrArray; //compiler error
    ptrArray = array; //no errors
}
```

pp Discussion.cpp Discussion array.cpp Untitled4

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

int main(){

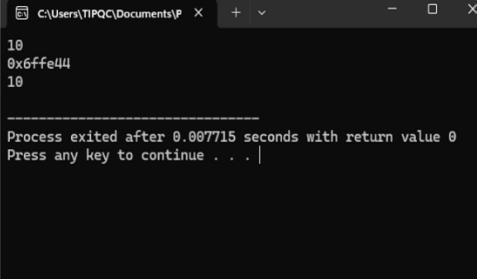
    int array[] = {1, 2, 3, 4};
    int *ptrArray;
    //array = ptrArray; //compiler error
    ptrArray = array; //no errors

    return 0;
}
```

### Part B.

test1.cpp PartB.cpp

```
#include <iostream>
using namespace std;
int main (){
    int var = 10;
    int *ip;
    ip = &var;
    cout << var << endl;
    cout << ip << endl;
    cout << *ip << endl;
    return 0;
}
```



### Part C.

DynamicAllocation.cpp

```
#include <iostream>
using namespace std;
int main(){
    int size = 5;
    int *array = new int[size];
    for (int i = 0; i < size; ++i) {
        array[i] = (i + 1) * 10;
    }
    std::cout << "Dynamically allocated array values:" << std::endl;
    for (int i = 0; i < size; ++i) {
        std::cout << "array[" << i << "] = " << array[i] << std::endl;
    }
    delete [] array;
    std::cout << "Memory for the array has been deallocated!" << std::endl;
    return 0;
}
```



## Part D.

The screenshot shows the Code::Blocks IDE interface. On the left, the code editor displays a C++ file named `main.cpp`. The code defines a `Student` class with a private member `obj_name` and a public member function `display_name` that prints the student's name. In the `main` function, a `Student` object `a` is created using dynamic memory allocation (`new`), and its `display_name` method is called. Finally, `a` is deleted. The build bar at the bottom indicates a successful build. On the right, the terminal window shows the output of the program: "Accessing data member through pointer: Joshua" and "Process exited after 0.09994 seconds with return value 0".

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 class Student {
6 public:
7     string obj_name;
8
9     Student(string name = "John Doe") {
10         obj_name = name;
11     }
12
13     void display_name() {
14         cout << "Student Name: " << obj_name << endl;
15     }
16 };
17
18 int main() {
19     Student *a = new Student("Joshua");
20
21     cout << "Accessing data member through pointer: " << (*a).obj_name << endl;
22
23     (*a).display_name();
24
25     delete a;
26
27     return 0;
28 }
29
```

## **Procedure:**

The screenshot shows a C++ code editor and its corresponding terminal window.

**Code (1.cpp):**

```
1.cpp
1 #include <iostream>
2 #include <string.h>
3
4 class Student{
5 private:
6 std::string studentName;
7 int studentAge;
8
9 public:
10
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
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 int main() {
34     Student student1("Roman", 28);
35     Student student2(student1);
36     Student student3;
37     student3 = student2;
38     return 0;
39 }
```

**Terminal Output:**

```
C:\Users\TIPQC\Documents\1. x
Constructor Called.
Copy Constructor Called
Constructor Called.
Destructor Called.
Destructor Called.
Destructor Called.

Process exited after 0.01097 seconds with return value 0
Press any key to continue . . .
```

```

cpp
1 #include <iostream>
2 #include <string.h>
3
4 class Student{
5 private:
6     std::string studentName;
7     int studentAge;
8
9 public:
10
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
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
35 int main(){
36     const size_t j = 5;
37     Student studentList[j] = {};
38     std::string namesList[j] = {"Carly", "Freddy", "Sam", "Zack", "Cody"};
39     int ageList[j] = {15, 16, 18, 19, 16};
40     return 0;
41 }

```

Output window:

```

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 0.008833 seconds with return value 0
Press any key to continue . . .

```

```

cpp
1 #include <iostream>
2 #include <string.h>
3
4 class Student{
5 private:
6     std::string studentName;
7     int studentAge;
8
9 public:
10
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
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
35 int main() {
36     const size_t j = 5;
37     Student studentList[j] = {};
38     std::string namesList[j] = {"Carly", "Freddy", "Sam", "Zack", "Cody"};
39     for(int i = 0; i < j; i++){ //Loop A
40         Student *ptr = new Student(namesList[i], ageList[i]); studentList[i]
41     }
42     for(int i = 0; i < j; i++){ //Loop B studentList[i].printDetails();
43     }
44     return 0;
45 }

```

Output window:

```

Constructor Called.

-----
Process exited after 0.01065 seconds with return value 0
Press any key to continue . . .

```

## 7. Supplementary Activity

The screenshot shows a C++ code editor with a file named "Data supp.cpp". The code defines a class "GroceryItem" with protected members: name (std::string), price (double), and quantity (int). It includes a constructor that initializes these members, a destructor, a virtual show() method that prints the item details, a virtual getTotal() method that calculates the total price, and a getName() method. The code then creates three instances of "GroceryItem": Apple (PHP 10, quantity 7), Banana (PHP 10, quantity 8), and Broccoli (PHP 60, quantity 12). It prints the grocery list, calculates the total price (PHP 1370), removes Lettuce from the list, and prints the updated list.

```
1 #include <iostream>
2 #include <string>
3
4
5 class GroceryItem {
6 protected:
7     std::string name;
8     double price;
9     int quantity;
10
11 public:
12     GroceryItem(std::string n = "", double p = 0.0, int q = 0)
13     {
14         name = n;
15         price = p;
16         quantity = q;
17     }
18
19     virtual ~GroceryItem() {}
20
21     virtual void show()
22     {
23         std::cout << name << " - PHP " << price << " x" << quantity
24         << " = PHP " << price * quantity << std::endl;
25     }
26
27
28     virtual double getTotal()
29     {
30         return price * quantity;
31     }
32
33     std::string getName()
34     {
35         return name;
36     }
37 }
38
39
```

## 8. Conclusion

So in this exercise, I was able to delve further into how arrays, pointers, and memory function in C++ in general, the entire idea of using new and delete to allocate dynamic memory. Working through it really gave me a sense of how the program was operating, and doing additional things like deleting an object or printing out the total price made it all the more hands-on and applicable. The entire process really showed me the step-by-step reasoning of programming, and that extra material like Jenna's Grocery List made it come so much more alive and interactive. I think I did fairly well although it's a bit hard given my situation but I'll definitely need to practice tracing the code better and keeping my memory management nice and tidy. This exercise seriously taught me how data is stored and manipulated in more complex programs.

## 9. Assessment Rubric