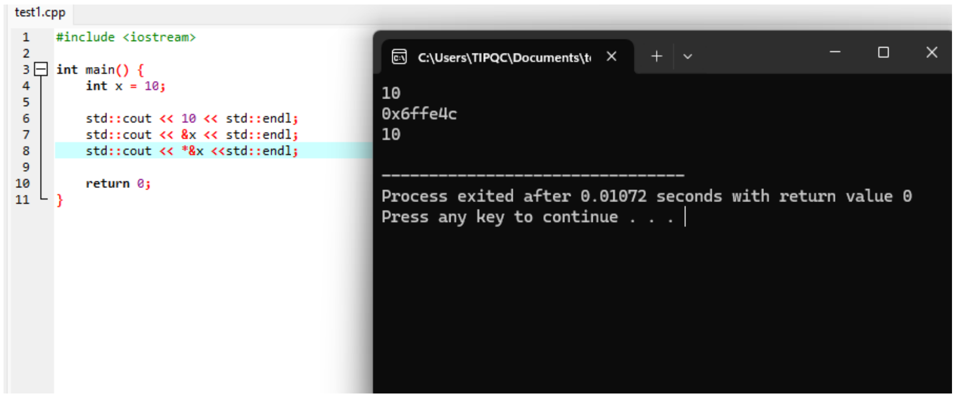
Activity No. <2>	
<ARRAYS, POINTERS AND DYNAMIC MEMORY ALLOCATION>	
Course Code: CPE010	Program: Computer Engineering
Course Title: Data Structures and Algorithms	Date Performed: 8/7/25
Section: CPE21S4	Date Submitted: 8/7/25
Name(s): Alcantara, Jason P.	Instructor: Engr. Jimlord Quejado

6. Output:

Discussion:

Part A.



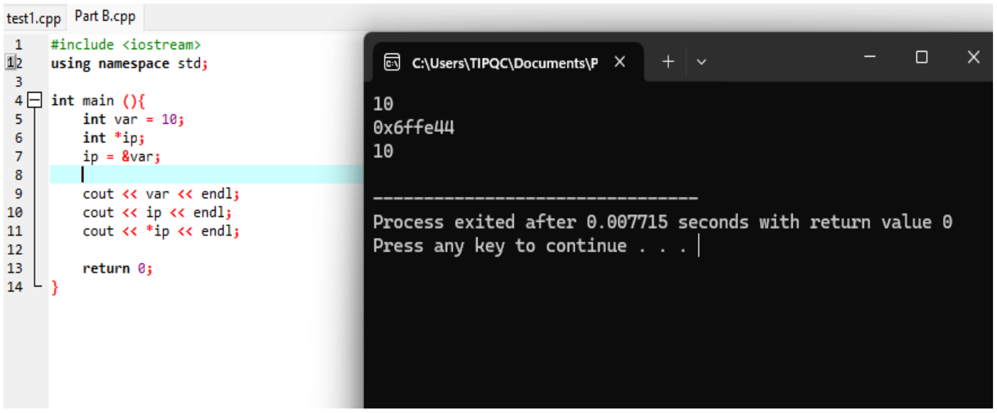
test1.cpp

```
#include <iostream>

int main() {
    int x = 10;

    std::cout << 10 << std::endl;
    std::cout << &x << std::endl;
    std::cout << *&x <<std::endl;

    return 0;
}
```

C:\Users\TIPQC\Documents\t

```
10
0x6ffe4c
10

--------------------------------
Process exited after 0.01072 seconds with return value 0
Press any key to continue . . .
```

Part B.



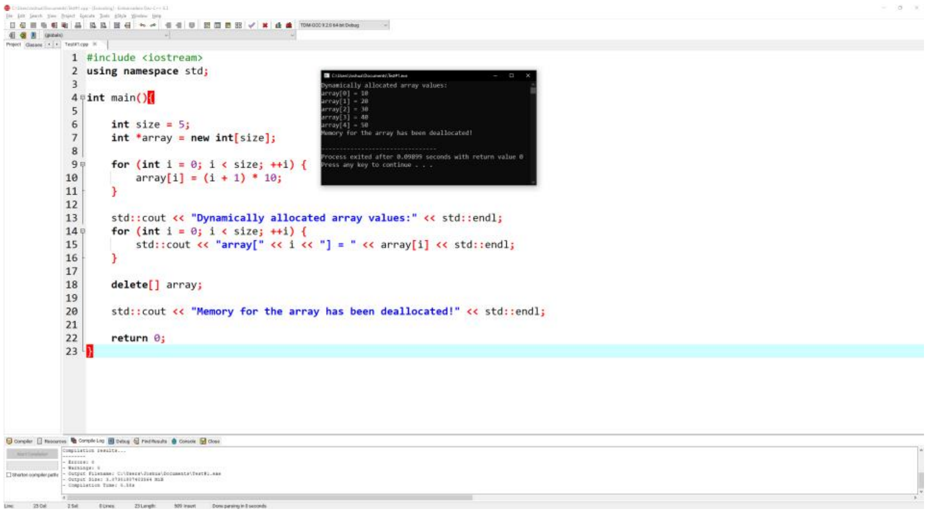
test1.cpp Part B.cpp

```
#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;
}
```

C:\Users\TIPQC\Documents\P

```
10
0x6ffe44
10

--------------------------------
Process exited after 0.007715 seconds with return value 0
Press any key to continue . . .
```

Part C.



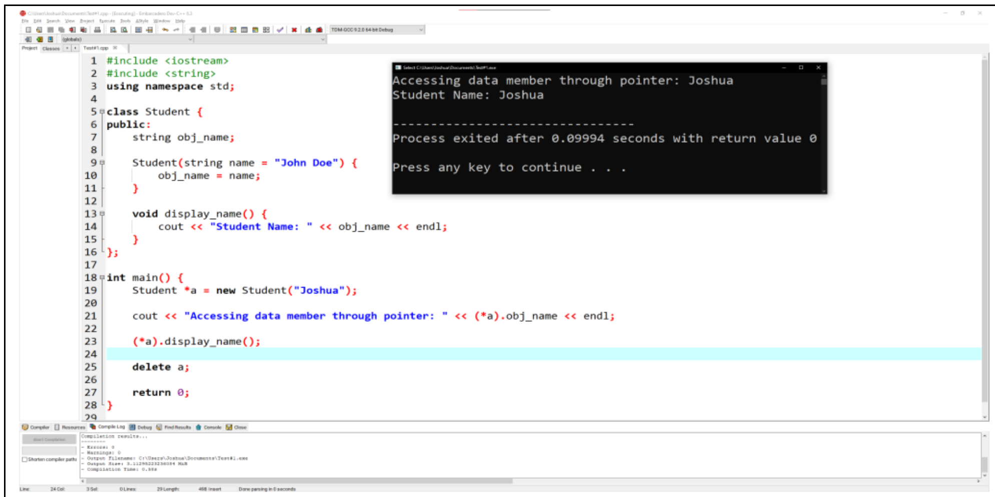
Test#1.cpp

```
#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;
}
```

```
Dynamically allocated array values:
array[0] = 10
array[1] = 20
array[2] = 30
array[3] = 40
array[4] = 50
Memory for the array has been deallocated!

--------------------------------
Process exited after 0.09899 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Joshua\Documents\Test#1.exe
- Output Size: [illegible] MiB
- Compilation Time: [illegible]
```

Part D.



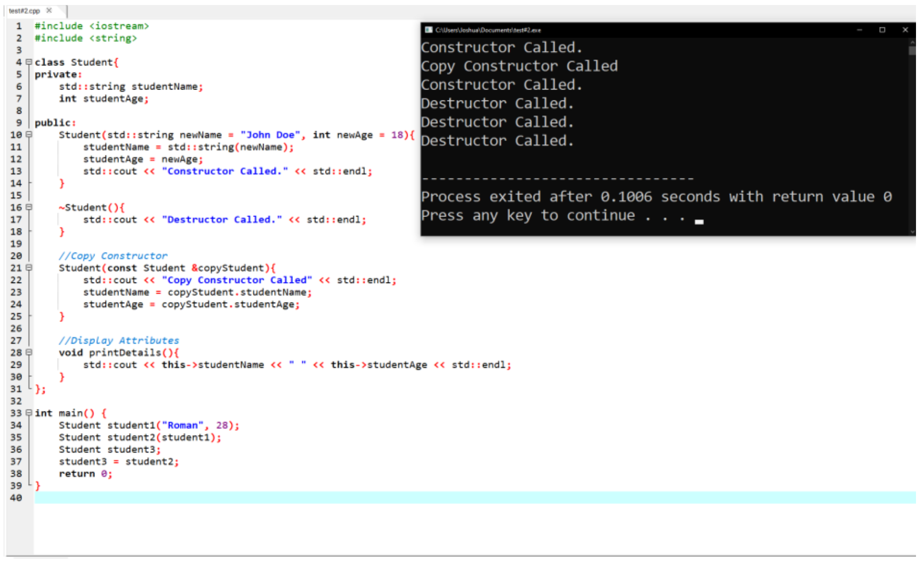
```cpp
#include <iostream>
#include <string>
using namespace std;

class Student {
public:
    string obj_name;

    Student(string name = "John Doe") {
        obj_name = name;
    }

    void display_name() {
        cout << "Student Name: " << obj_name << endl;
    }
};

int main() {
    Student *a = new Student("Joshua");

    cout << "Accessing data member through pointer: " << (*a).obj_name << endl;

    (*a).display_name();

    delete a;

    return 0;
}
```

```
Accessing data member through pointer: Joshua
Student Name: Joshua

--------------------------------
Process exited after 0.09994 seconds with return value 0

Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Joshua\Documents\Test#1.exe
- Output Size: 3.11295223236084 MiB
- Compilation Time: 0.38s
```

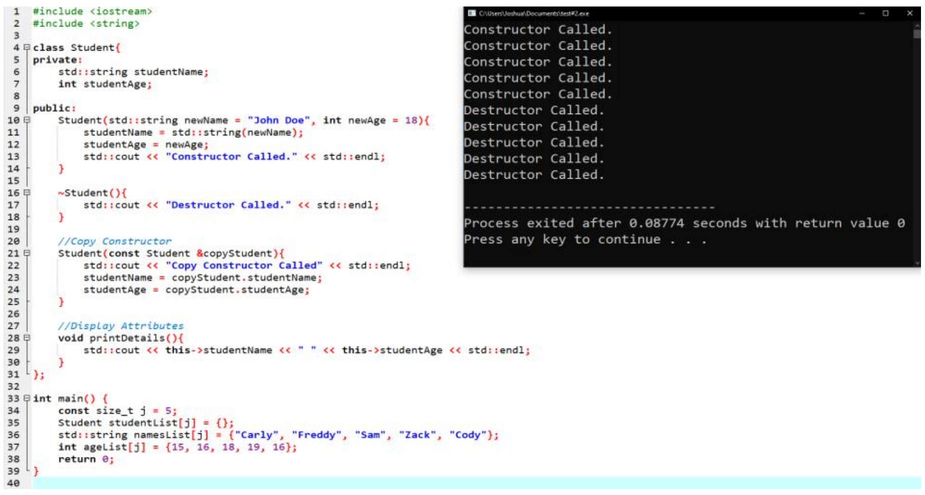
Procedure:

ILO A:



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

class Student{
private:
    std::string studentName;
    int studentAge;

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

    ~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;
}
```

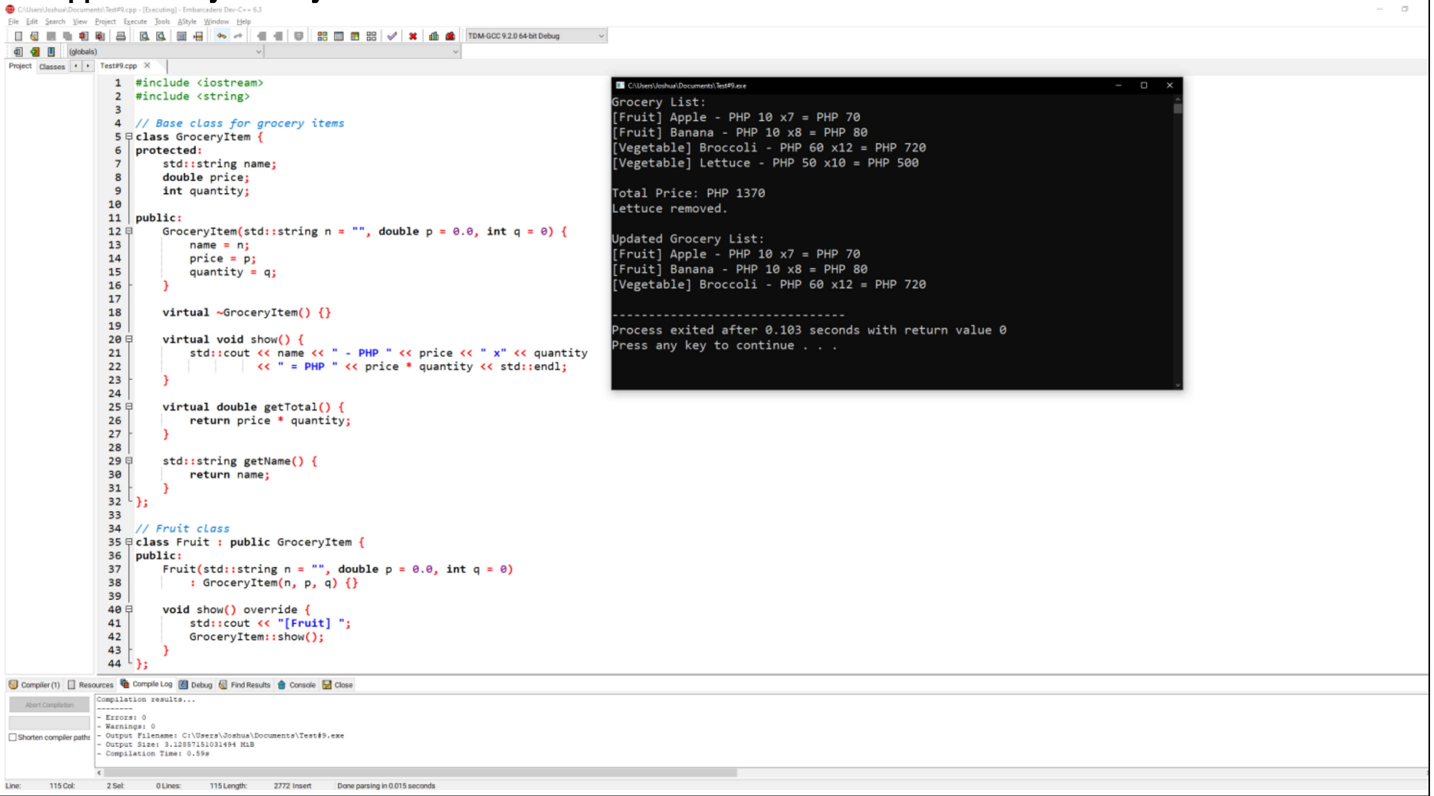
```
Constructor Called.
Copy Constructor Called
Constructor Called.
Destructor Called.
Destructor Called.
Destructor Called.


--------------------------------
Process exited after 0.1006 seconds with return value 0
Press any key to continue . . .
```



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

class Student{
private:
    std::string studentName;
    int studentAge;

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

    ~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() {
    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.

--------------------------------
Process exited after 0.08774 seconds with return value 0
Press any key to continue . . .
```

7. Supplementary Activity:



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

// Base class for grocery items
class GroceryItem {
protected:
    std::string name;
    double price;
    int quantity;

public:
    GroceryItem(std::string n = "", double p = 0.0, int q = 0) {
        name = n;
        price = p;
        quantity = q;
    }

    virtual ~GroceryItem() {}

    virtual void show() {
        std::cout << name << " - PHP " << price << " x" << quantity
                  << " = PHP " << price * quantity << std::endl;
    }

    virtual double getTotal() {
        return price * quantity;
    }

    std::string getName() {
        return name;
    }
};

// Fruit class
class Fruit : public GroceryItem {
public:
    Fruit(std::string n = "", double p = 0.0, int q = 0)
        : GroceryItem(n, p, q) {}

    void show() override {
        std::cout << "[Fruit] ";
        GroceryItem::show();
    }
};
```

```
Grocery List:
[Fruit] Apple - PHP 10 x7 = PHP 70
[Fruit] Banana - PHP 10 x8 = PHP 80
[Vegetable] Broccoli - PHP 60 x12 = PHP 720
[Vegetable] Lettuce - PHP 50 x10 = PHP 500

Total Price: PHP 1370
Lettuce removed.

Updated Grocery List:
[Fruit] Apple - PHP 10 x7 = PHP 70
[Fruit] Banana - PHP 10 x8 = PHP 80
[Vegetable] Broccoli - PHP 60 x12 = PHP 720

--------------------------------
Process exited after 0.103 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Joshua\Documents\Test#9.exe
- Output Size: 3.12857151031494 MiB
- Compilation Time: 0.59s
```

8. Conclusion

- **Summary of lessons learned:** I understand how the dynamic memory allocation works. This helps manage memory more organize and understandable, especially for the data with unknown size.
- **Analysis of the procedure:** In this procedure you need to familiarize to the functions like (malloc, free and etc.), and how you applied them to manage memory in this codes.
- **Analysis of the supplementary activity:** The more activities and practices that you did it will helped us to reinforce the lesson, such as understanding the memory management in the real life world programming scenarios.
- **Concluding statement / Feedback:** How well did you think you did in this activity? What are your areas for improvement?: I think I need to practice more to just understand the dynamic memory allocated because I still get confused because so that's why I need to improve myself in this program.

9. Assessment Rubric