Assignment 4.3	
Pointers	
Course Code: CPE 007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: September 18, 2025**
Section: CPE11S1	**Date Submitted: September 18, 2025**
Name(s): Jaime Luis M. Demain	**Instructor: Engr. Jimlord M. Quejado**

6. Output

1. **What is a pointer in C++? Pointer is a special variable that stores the memory address of another variable. It "points" where the value is stored in memory rather than holding it directly.**
2. **How does a pointer differ from a regular variable? Regular Variable stores data directly while Pointer stores a memory address of a data, like holding its address, thus only accessing data through dereferencing.**
3. **What operator is used to get the address of a variable?**

   Answer: &
4. **What operator is used to access the value stored at a pointer's address?**

   Answer: *
5. **Why are pointers important in C++? Give two uses. It is important as it allows you to directly manipulate the memory, thus giving you more efficiency on your programs. Here are the two uses of pointers:**

   First example:

   ```
   int num[] = {4, 5, 6};
   int* n = num; // p points to num[0]
   cout << *(n + 1); // Accesses num[1]
   ```

   Second example:

   ```
   void changeValue(int* ptr) {
       *ptr = 10;
     }
   // In main:
   int num = 5;
   changeValue(&num); // num is now 10
   ```

7. Supplementary Activity

1.
   ```
   int x = 42;
   int *ptr = &x;
   cout << *ptr;
   ```

   Output: 42

   Analysis: It will output 42 as 42 is only the sole value of x.

2. int a = 5, b = 10;

   int *p = &a;

   p = &b;

   cout << *p;

   Output: 10

   Analysis: It will output 10 as the latest code implies that p is equal to the address of b.

3. int arr[3] = {10, 20, 30};
   int *p = arr;
   cout << *p;

   Output: 10

   Analysis: It will output 10 as the p (index of accessed value) is not incremented, assigned to 2nd above index, or doesn't contain mathematical operation in the cout to manipulate the pointed index of an array.

4. int arr[4] = {2, 4, 6, 8};

   int *p = arr;

   p++;

   cout << *p;

   Output: 4

   Analysis: It will output 4 as the p (index of accessed value) is incremented and the pointed index number will increase by 1 (0 index to 1 index).

5. int arr[3] = {5, 15, 25};

   int *p = arr;

   cout << *(p + 2);

   Output: 25

   Analysis: It will output 25 as the p (the index) is added by 2, thus making the accessed index to 2 (3rd index) and it will print out 25.

Error Spotting:

1. int arr[3] = {1, 2, 3};

   int *p = &arr;

   New Version:

   int arr[3] = {1, 2, 3};

   int *p = &arr;

   cout << *p;

   Analysis: It doesn't contain **cout** to print it out.

2. int arr[5];

   int *p;

   p = arr[2];

   New Version:

   int arr[5];

   int *p;

   p = &arr[2];

   cout << *p;

   Analysis: It doesn’t contain “&” at the third line when p is assigned to that specific index of an array and also doesn’t include **cout**.

3. int arr[4] = {10, 20, 30, 40};

   cout << *arr[2];

   New Version:

   int arr[4] = {10, 20, 30, 40};
   cout << *(arr + 2);

   Analysis: When inputting a specific index that has a pointer, you must use parenthesis, place the array name there, and add by the number that will result on printing out the specific index that you want to print out.

8. Conclusion

This activity is basically recalling about what’s the pointer, its operators, and its uses plus the supplementary activities. Supplementary activities only do identifying the outputs in pointers and error fixing. What I learned in this activity is that when you want to print out the specific index that has a pointer, you must use “&” when directly instructing to display that particular index (arr[2] for example), use increments to increase the assigned index to be printed out, or use parenthesis with addition operator (*(arr + 2) for example). I also learned that pointers can optimize your program’s performance by having direct control over your memory. And lastly, I learned the difference between the regular variable and the pointers. Pointers can only “locate” the value while regular variable store the value.