Assignment 4.3	
Pointers	
Course Code: CPE 007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: 9/23/2025**
Section: CPE11S1	**Date Submitted: 9/23/2025**
Name(s): Juan Paulo C. Lara	**Instructor: Engr. Jimlord M. Quejado**

6. Output

1. **Pointer is a variable operator storing the actual memory address contained by a variable. It stores the location where that memory is allocated.**
2. **A regular variable stores the given value as is. A pointer variable stores the memory address of a variable.**
3. **To get the address of a variable, use the address operator (&variable).**
4. **To access the value stored at a pointer's address, use the dereference operator (*variable).**
5. **Pointers are important because they allow programs to directly access and control computer memory. Example uses include managing allocated memory during execution and control flow and low level programming for embedded systems.**

7. Supplementary Activity

Identify the Output
For each code snippet, predict the output without compiling:

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

Output: 42

2.
```
int a = 5, b = 10;
int *p = &a;
p = &b;
cout << *p;
```

Output: 10

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

Output: 10

4.
```
int arr[4] = {2, 4, 6, 8};
int *p = arr;
p++;
cout << *p;
```

Output: 4

5.
```
int arr[3] = {5, 15, 25};
int *p = arr;
cout << *(p + 2);
```

Output: 25

Error Spotting
Identify and fix the error(if any) in the codes below.

1.
```
int arr[3] = {1, 2, 3};
int *p = &arr;
```

Error: Line 2. Cannot assign a memory address of an array to int pointer.
Fixed:
```
int arr[3] = {1, 2, 3};
int *p = arr;
```

2.
```
int arr[5];
int *p;
p = arr[2];
```

Error: Line 3. Invalid assignment of pointer p to int p.
Fixed:
```
int arr[5];
int *p;
p = &arr[2];
```

3.
```
int arr[4] = {10, 20, 30, 40};
cout << *arr[2];
```

Error: Line 2. Since arr[2] is int, a dereference operator is not needed.
Fixed:
```
int arr[4] = {10, 20, 30, 40};
cout << arr[2];
```

8. Conclusion

Concluding this activity, I learned that pointers allow for direct access and control of program memory when running it. Its operators, &address and *dereference allows manipulation of memory without the cost of initializing, storing, and changing the contained values directly. Because it only contains the memory address instead of the variable itself, programs, especially more complex and larger ones, can compile and run faster. Even if the memory addresses are different computer to computer, variables and addresses are still essentially the same thing to the computer that is running the program. With this amount of control given to a programmer, operations should be checked in code to avoid invalid entries, errors, and illegal operations that can cause exceptions and crashes that may be detrimental to a system like servers and computer clusters. Overall, this and the previous activities gave me a deeper understanding of how this language interacts with computers, more specifically its memory.