Activity No. 5.2	
Hands-on Activity 5.2: Structures	
Course Code: CPE007	**Program: Computer Engineering**
Course Title: Programming Logic and Design	**Date Performed: October 3, 2025**
Section: CPE11S1	**Date Submitted: October 5, 2025**
Name(s): Tobias, Lawrence C.	**Instructor: Engr. Jimlord M. Quejado**

6. Output

1.

The code is about making a “Card” that has a face and a suit, then showing those on the screen. It starts by creating a structure called “Card” that has two parts named “face” and “suit”. In the main part, there’s a variable called “Card a” and also a pointer called “Card* aPtr”. The code then gives “Card a” the values “Ace” for the face and “Spades” for the suit, so it basically represents the Ace of Spades card from a deck. After that, it shows how to print the values of that card in three different ways. It uses the dot operator, the arrow operator, and the dereference operator to print the same thing. The main idea is to show that there are many ways to get or show data from a structure, and this code helps you understand how each one works.

2.

For number 2 it is about using a structure to store and organize details about books. It starts by creating a structure called “Book” that has four parts which are “title”, “author”, “subject”, and “book_id”. After that, the code makes two book variables named “Book1” and “Book2”. Each of these can hold one book’s details, like its title and author. Then the program gives each book its own information. For example, Book1 has its own title, author, subject, and ID number, and Book2 also has different details. After filling in the information, the code prints all of them to the screen so you can see how it looks. This shows that using structures makes it easy to keep related information together instead of using separate variables. It makes the program more organized and easier to understand because everything about a book is inside one structure.

3.

Number 3 is like an improved version of number 2 because it makes the code neater by using a function to print the book details. Instead of printing everything one by one in the main part, it creates a function that takes the book details and prints them all together. That way, the code looks cleaner, shorter, and easier to edit. For example, if you want to print another book later, you don’t have to write all the print lines again. You can just call the function and it will show all the details for you. It also helps make the program look more organized since the printing part is separated from the main part of the code. Overall, it’s a better version because it shows how using functions can make your program easier to read and reuse, especially when you’re dealing with structures.

7. Supplementary Activity

1.

[*] SECRETT.cpp NO CLUEE.cpp

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

struct Rectangle {
    double length;
    double width;
};

void computeRectangle(Rectangle r) {
    double area = r.length * r.width;
    double perimeter = 2 * (r.length + r.width);

    cout << "Length: " << r.length << endl;
    cout << "Width : " << r.width << endl;
    cout << "Area  : " << area << endl;
    cout << "Perimeter: " << perimeter << endl;
}

int main() {
    Rectangle rect;

    cout << "Enter the length of the rectangle: ";
    cin >> rect.length;
    cout << "Enter the width of the rectangle: ";
    cin >> rect.width;

    computeRectangle(rect);

    return 0;
}

```

C:\Dev-Cpp\SECRETT.exe

```
Enter the length of the rectangle: 13
Enter the width of the rectangle: 4
Length: 13
Width : 4
Area  : 52
Perimeter: 34

--------------------------------
Process exited after 5.186 seconds with return value 0
Press any key to continue . . .
```

- This code is about finding the area and perimeter of a rectangle. It starts by including the iostream library so it can use cout and cin. Then it makes a structure called Rectangle with two parts: length and width. There's also a function called computeRectangle that takes a rectangle, finds the area by multiplying length and width, and gets the perimeter by adding them and multiplying by 2. The function then prints the length, width, area, and perimeter. In the main part, it makes a rectangle variable named rect, asks the user to enter the length and width, and saves them. After that, it calls the function to do the calculations and show the results.

2.

[*] SECRETT.cpp | NO CLUEE.cpp

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


bool multiple(int num, int x) {
    if (num % x == 0) {
        return true;
    } else {
        return false;
    }
}

int main() {
    int num, x;


    cout << "Enter a number: ";
    cin >> num;
    cout << "Enter the value of x: ";
    cin >> x;

      if (multiple(num, x)) {
        cout << num << " is a multiple of " << x << endl;
    } else {
        cout << num << " is NOT a multiple of " << x << endl;
    }

    return 0;
}

```

```
C:\Dev-Cpp\NO CLUEE.exe
Enter a number: 13
Enter the value of x: 4
13 is NOT a multiple of 4

--------------------------------
Process exited after 100.6 seconds with return value 0
Press any key to continue . . .
```

```
C:\Dev-Cpp\NO CLUEE.exe
Enter a number: 15
Enter the value of x: 3
15 is a multiple of 3

--------------------------------
Process exited after 4.399 seconds with return value 0
Press any key to continue . . .
```

- This code checks if one number is a multiple of another. It starts by including the iostream library so the program can use cout and cin. Then there's a function called multiple that takes two numbers, num and x, and checks if num divided by x leaves no remainder using the % symbol. If the remainder is 0, it

returns true; if not, it returns false. In the main part, the program asks the user to enter two numbers, then uses the function to check if the first number is a multiple of the second. If it is, it prints “is a multiple of x,” and if not, it prints “is NOT a multiple of x.” Finally, it ends with return 0 to show the program ran successfully.

8. Conclusion

- While doing all these activities, I learned a bunch of stuff about using structures and functions in C++. In the first program, I learned how to use a structure to store a card’s face and suit and learned how to print it in different ways. Then in the next one, I learned how to use structures again to save book details like the title and author, and later improved it by adding a function to print them more neatly. In the rectangle part, I learned how to use both a structure and a function to find the area and perimeter based on what the user enters. And for the last one, I learned how to use a code that checks if one number is a multiple of another using a function and if-else statements. Doing all these really helped me understand how functions and structures work together. I think I’m improving little by little, but I still need more practice so I can code faster and make fewer mistakes.
-