Hands-on Activity 5.2:	
Structures	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:** 10/4/2025
Section: CPE11S1	**Date Submitted:** 10/5/2025
Name(s): Ramirez, Angel Mae C.	**Instructor:** Engr. Jimlord M. Quejado

6. Output

1. CODE ANALYSIS:

From this code, I observed how a structure named Card is used to store the face and suit of a playing card. The structure variable a was used to assign values, and a pointer aPtr was used to show how to access members using different methods. What stood out to me was the three different ways the program accessed the structure members: directly using the dot operator (a.face), using the arrow operator with a pointer (aPtr->face), and by dereferencing the pointer ((*aPtr).face). All of them gave the same output, but I understood that the arrow operator is specifically used with structure pointers, which makes accessing members easier.

2. CODE ANALYSIS:

In this part of the lesson, I saw how a structure called Books was used to store information like the book's title, author, subject, and ID. The code declared two structure variables, Book1 and Book2, and then assigned different values to each of them. After assigning the values, the code printed out the information using cout. I noticed how each structure instance held its own data and how easy it was to access each member using the dot operator. This helped me understand how structures can organize related data in a clean and reusable way.

3. CODE ANALYSIS:

This example showed how a structure can be passed to a function. The printBook function accepted a Books structure as an argument and printed out all the book information. I observed that the structure was passed by value, meaning the function got a copy of the original data. This means that even if the function tried to modify the structure, it wouldn't affect the original in main. This example helped me see how structures can be passed into functions to make code more organized and reusable, especially when dealing with larger or grouped data like records.

7. Supplementary Activity

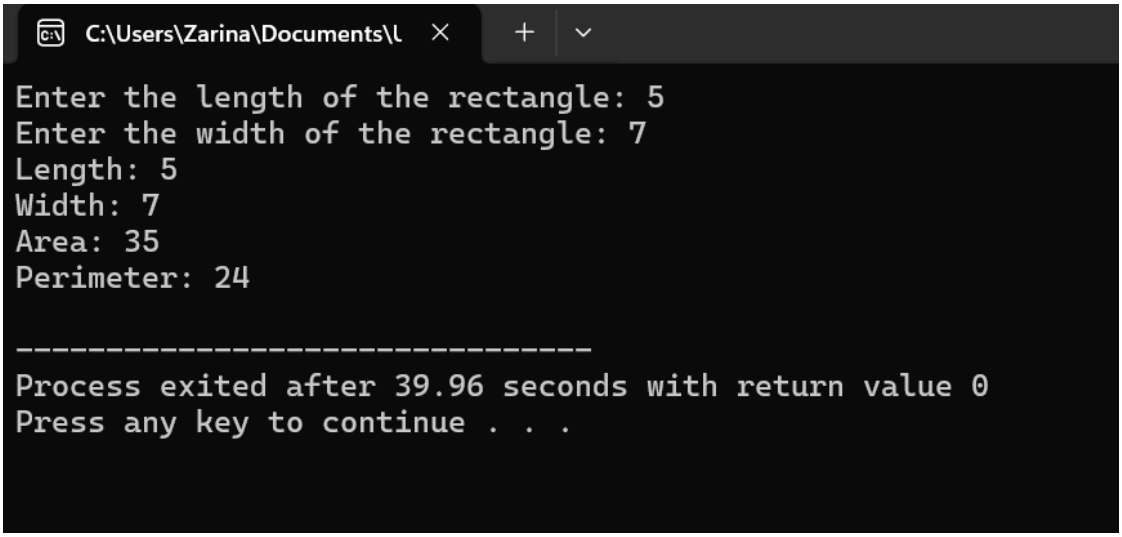
1.Create a program that uses a structure to store a rectangle's length and width. Write a function that accepts the structure as an argument and computes the area and perimeter of the rectangle.

CODE:

Untitled1.cpp

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


struct Rectangle {
    double length;
    double width;
};


void compute(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;


    compute(rect);

    return 0;
}

```

OUTPUT:



ANALYSIS:

In this program, I created a structure called Rectangle that stores two double variables: length and width. I used this structure to group together the measurements of a rectangle. In the main() function, I asked the user to input the length and width, and stored those values inside the structure variable rect. I then passed this structure to a function named computeAreaPerimeter. This function takes the structure as a parameter (passed by value) and calculates the area using length * width, and the perimeter using 2 * (length + width). It then displays the results. By writing this program, I practiced how to define a structure, how to collect input from the user into that structure, and how to pass the entire structure to a function to perform calculations and output the results.

2. Write a program that creates a function multiple that determines if the integer entered from a keyboard is a multiple of some integer x.

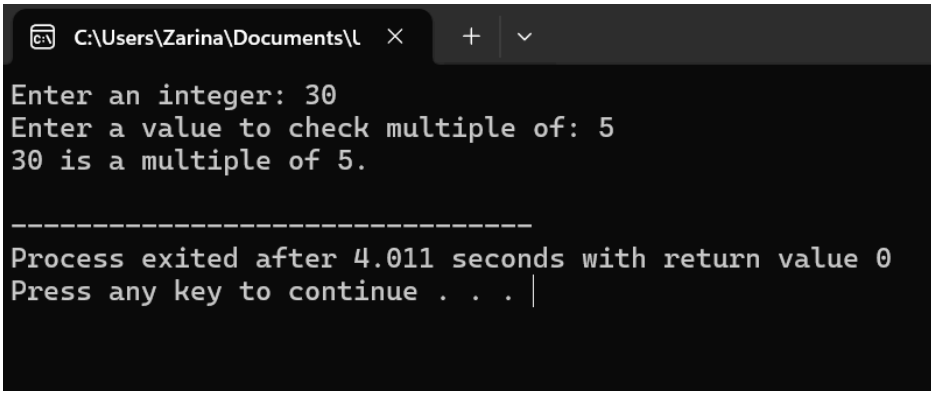
CODE:

Untitled1.cpp

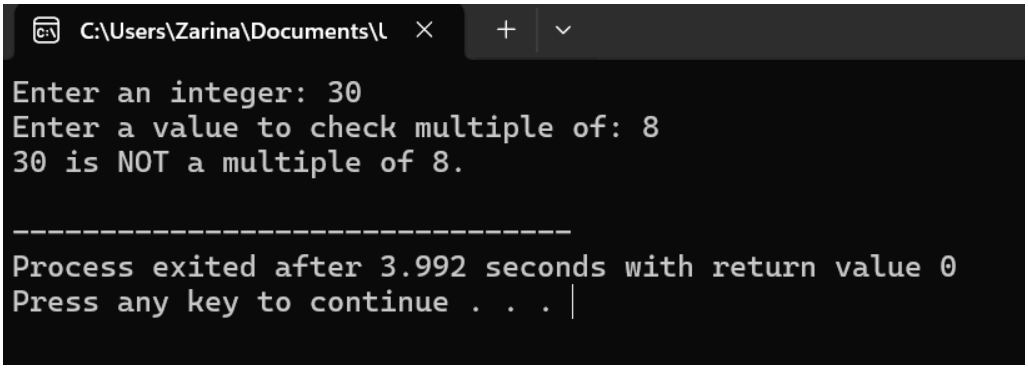
```cpp
#include <iostream>
using namespace std;


bool multiple(int number, int x) {
    return (number % x == 0);
}

int main() {
    int number, x;


    cout << "Enter an integer: ";
    cin >> number;
    cout << "Enter a value to check multiple of: ";
    cin >> x;


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

    return 0;
}

```

OUTPUT:

IS A MULTIPLE



NOT A MULTIPLE



ANALYSIS:

For this activity, I wrote a program that checks if one number is a multiple of another. I created a function called multiple that takes two integers as parameters: the number to check, and the number to check against. Inside the function, I used the modulo operator (%) to check if the remainder of num % x is 0, which would mean the first number is a multiple of the second. The function returns true or false depending on the result. In the main() function, I asked the user to enter two numbers, called the multiple function with those numbers, and then used an if statement to display whether the first number is a multiple of the second. By doing this, I practiced writing a function that returns a boolean, working with conditionals, and using the modulo operator to solve a simple mathematical problem.

8. Conclusion

Through this activity, I learned how to create and use structures in C++ to group related variables together. I realized that structures are very useful when handling complex data types, especially when organizing multiple pieces of information under one name. I was able to define structure variables, assign values to their members, and access those members using both the dot operator and the arrow operator with pointers. I also practiced passing structures to functions by value, which helped me understand how data is copied and used inside functions without changing the original structure. In the first program, I applied this knowledge to calculate the area and perimeter of a rectangle using a structure, while in the second program, I created a function to check if a number is a multiple of another using basic logic and return values. Overall, this activity improved my understanding of structures, function calls, and how to organize data more efficiently in C++ programming.