Activity No. n 5.2	
Hands-on Activity 5.2 Structure	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:09/30/25**
Section: CPE11S1	**Date Submitted:10/04/25**
Name(s):Cenndy M. Nieles	**Instructor: Engr. Jimlord M. Quejado**

6. Output

SAMPLE #1

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

struct Card {
   string face;
   string suit;
};

int main() {
   Card a;       // structure variable
   Card* aPtr;   // structure pointer

   // Assign values
   a.face = "Ace";
   a.suit = "Spades";

   // Pointer points to structure 'a'
   aPtr = &a;

   // Accessing members:
   //  Using the dot operator (.)
   cout << a.face << " of " << a.suit << endl;

   //  Using the arrow operator (->)
   cout << aPtr->face << " of " << aPtr->suit << endl;

   //  Using dereference (*) and dot
   cout << (*aPtr).face << " of " << (*aPtr).suit << endl;

   return 0;
}
```

EXPLANATION:

First includes the header to declare an object that is going to be included so the program can use string data types. Next it defined a structure name Card which has two members the string face and string suits the inside the main function structure pointer aPtr are declared .Then assign the values of a.face = “Ace”; and a. suits = “Spades”; then set the address which means the pointer is now pointed to the structure.As a result, the program is easier to administer and more structured. Secondly, it demonstrates the use of pointers with structures. One of C++'s most crucial features is

*pointers. The second approach makes use of the structure pointer aPtr and the arrow operator (->). The same values are printed using the pointer in the following statement: cout \< aPtr->face << " of " \< aPtr->suit << endl;. The arrow operator is a shortcut for using a pointer to access members of a structure. You can write aPtr->face in place of (*aPtr).face. They let you create dynamic data structures like linked lists, trees, and graphs, work directly with memory, and pass big data structures to functions without copying them. More complex programming starts with knowing how to use pointers with structures.*

SAMPLE#2

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

// Define the structure
struct Books {
    string title;
    string author;
    string subject;
    int book_id;
};

int main() {
    // Declare two Book variables
    Books Book1;
    Books Book2;

    // Book 1 specification
    Book1.title = "C Programming";
    Book1.author = "Nuha Ali";
    Book1.subject = "C Programming Tutorial";
    Book1.book_id = 6495407;

    // Book 2 specification
    Book2.title = "Telecom Billing";
    Book2.author = "Zara Ali";
    Book2.subject = "Telecom Billing Tutorial";
    Book2.book_id = 6495700;

    // Print Book 1 info
    cout << "Book 1 title   : " << Book1.title << endl;
    cout << "Book 1 author  : " << Book1.author << endl;
    cout << "Book 1 subject : " << Book1.subject << endl;
    cout << "Book 1 book_id : " << Book1.book_id << endl;

    cout << endl; // for spacing

    // Print Book 2 info
    cout << "Book 2 title   : " << Book2.title << endl;
    cout << "Book 2 author  : " << Book2.author << endl;
    cout << "Book 2 subject : " << Book2.subject << endl;
    cout << "Book 2 book_id : " << Book2.book_id << endl;
```

```
    return 0;
}
```

EXPLANATION:

This program demonstrates the use of a struct. The program defines a struct named Books to hold information about a book, including its title, author, subject, and a book ID. Inside the main function, two variables of this Books struct, Book1 and Book2, are declared. The program then assigns specific string and integer values to the members of each of these book variables. Finally, the code uses cout statements to print the information for both Book1 and Book2 to the console.Together, the four members of the Books structure—title, author, subject, and book_id—describe a book's specifics. Book1 and Book2, two variables of type Books, are declared in the main function. Each of these variables is then given a value by the program, with Book 1 containing information about a C programming book and Book 2 containing information about a telecom billing book. Following value assignment, the program uses cout to print the stored data from both books, including the ID, title, author, and subject, in an understandable format.

SAMPLE#3

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

// Define a structure
struct Books {
    string title;
    string author;
    string subject;
    int book_id;
};

// Function declaration (structure passed by value)
void printBook(Books book);

int main() {
    // Declare two Book variables
    Books Book1;
    Books Book2;

    // Book 1 specification
    Book1.title = "C Programming";
    Book1.author = "Nuha Ali";
    Book1.subject = "C Programming Tutorial";
    Book1.book_id = 6495407;

    // Book 2 specification
    Book2.title = "Telecom Billing";
    Book2.author = "Zara Ali";
    Book2.subject = "Telecom Billing Tutorial";
    Book2.book_id = 6495700;
```

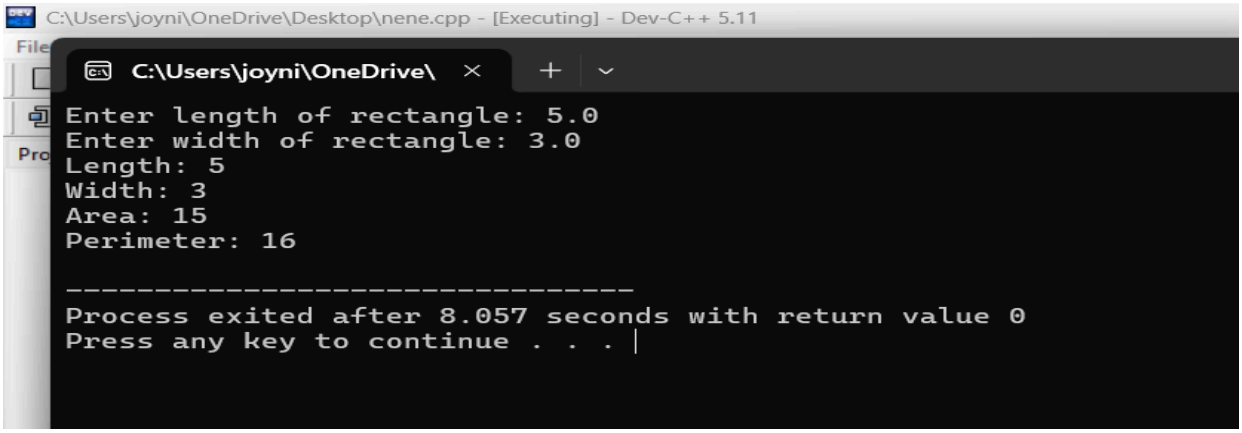
```
    // Print details by passing the structure to a function
    printBook(Book1);
    cout << endl; // just for spacing
    printBook(Book2);

    return 0;
}

// Function definition
void printBook(Books book) {
    cout << "Book title   : " << book.title << endl;
    cout << "Book author  : " << book.author << endl;
    cout << "Book subject : " << book.subject << endl;
    cout << "Book book_id : " << book.book_id << endl;
}
```

EXPLANATION:

The provided C programming code defines a Books structure containing a subject and a book ID. It declares a function printBook that accepts a Books structure. In the main function, two Books variables, Book1 and Book2, are created. Book1 is initialized with details for "C Programming" with a book ID of , and Book2 is initialized for "Telecom Billing" with a book ID .Inside main, two variables of type Books are declared: Book1 and Book2.The use of the printBook function that follows exemplifies modularity, a key concept in programming.The structure is defined using struct it means named books and books structure has four group member the title, author, subject , and book id the title stores the name of the book, the author stores the name of the person who wrote it, and the subject describes the topic of the book.

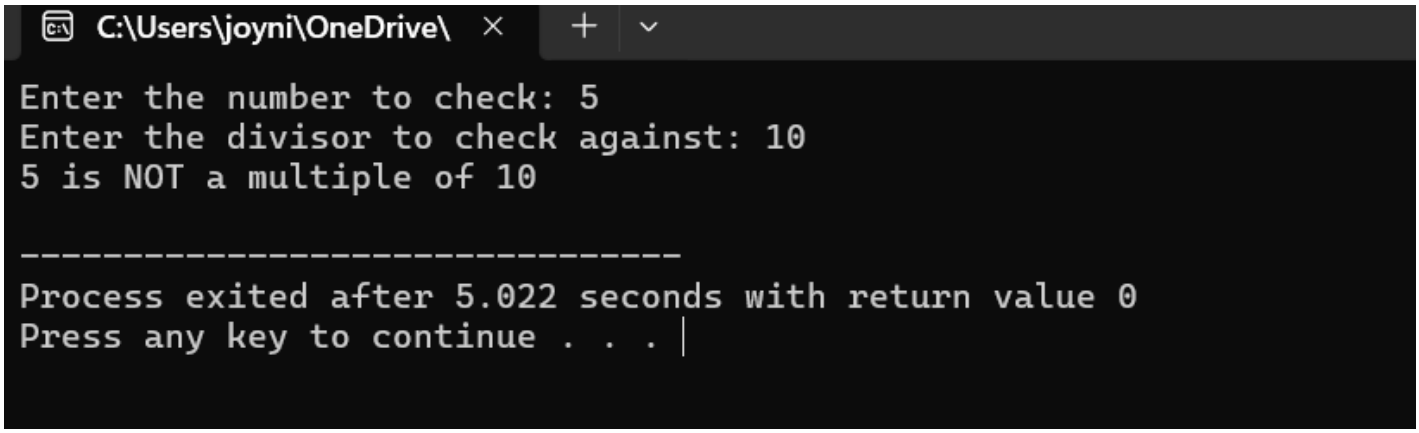
7. Supplementary Activity

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

// Define the structure for Rectangle
struct Rectangle {
    double length;
    double width;
};

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

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

int main() {
    Rectangle r;

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

    compute(r);

    return 0;
```



EXPLANATION:

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

// Define the structure for Rectangle
struct Rectangle {
    double length;
    double width;
};

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

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

int main() {
    Rectangle r;

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

    compute(r);

    return 0;
}
```



EXPLANATION:

The struct is called Rectangle to hold the length and width of a rectangle as a double variable .The program includes a function that takes a Rectangle object as input. Then calculate the area and perimeter of the the rectangle.The main function prompts the user to enter the length and width of a rectangle , stores these values in a rectangle object , and then compute function display the length,width,calculate area and parameter to console
This cpp program determines if a given number is a multiple of another. All of the primary logic is handled by a function called multiple(). The user-inputted numbers are stored in two integer variables, num and x, which are created inside the function. In order to determine the divisor, the program first asks the user to enter a number, which will be the primary number to be checked. Following input, the program determines whether the divisor x equals zero. Since dividing by zero is prohibited in both computers and mathematics, the program displays an error message if it does not.The program checks to see if the first number can be divided evenly by the second if the divisor is not zero. The program prints a message to verify that the first number is a multiple of the second if the remainder is zero. The program indicates that the number is not a multiple of the divisor if the remainder is not zero.

8. Conclusion

Using structures makes programs cleaner, easier to understand, and more manageable, since instead of handling many separate variables, you can work with a single structured variable that holds all the necessary information. Structures also enhance reusability, as once defined, they can be used to create multiple instances that store different sets of values. Furthermore, they can be passed as arguments to functions, returned from functions, or manipulated with pointers, giving programmers flexibility and control when handling data."structs" in C++ are a fundamental user-defined data type that allows programmers to group together variables of different data types under a single name, creating a composite data structure that can represent complex entities or records in a program. When declaring a structure, the `struct` *keyword precedes the user-chosen name, and the list of member variables is contained within curly braces, with a semicolon required to terminate the entire declaration.Structures allow the grouping of diverse data types int, string, float under a single name, enhancing code readability and manageability, especially when dealing with multiple instances of similar information.*