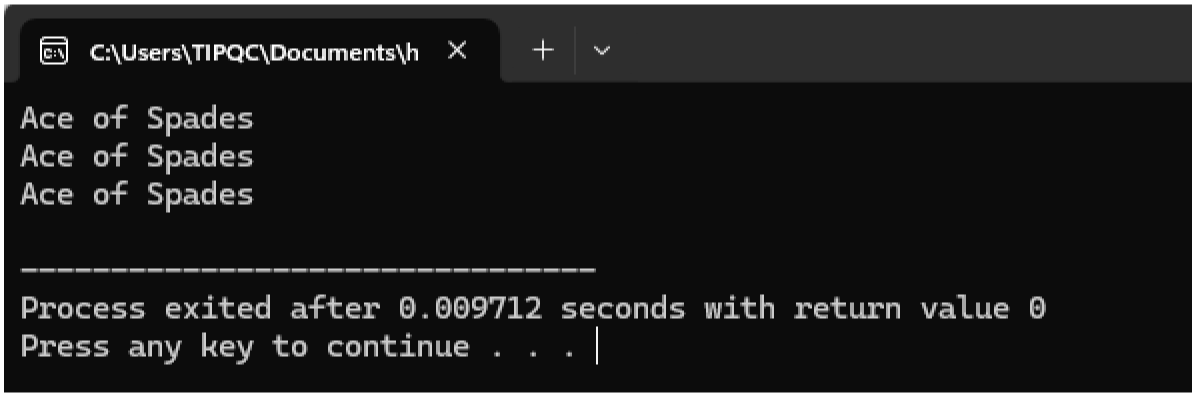
Activity No. 5.2	
Structures	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:9/30/25**
Section:CPE11S1	**Date Submitted:9/30/25**
Name(s): Will Stuart D. Ponce Jr.	**Instructor:** Engr. Jimlord M. Quejado

6. Output:
Sample Code:

```
#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;
}
```

Analysis:
Essentially, the program is a visual exemplification of the data-structure 'structure' in C++ a model that represents the relationships between the 'face' and 'suit' of a playing card.Besides that, it additionally displays the three ways of the structure members' access i.e. via the dot operator, the arrow operator together with the pointer and the de referencing with the dot.
Code:



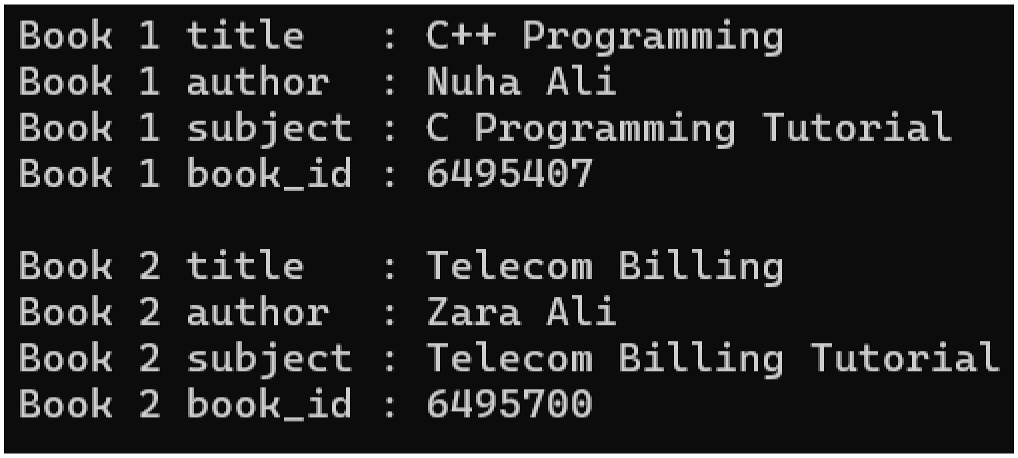
```
Ace of Spades
Ace of Spades
Ace of Spades

--------------------------------
Process exited after 0.009712 seconds with return value 0
Press any key to continue . . .
```

```
#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;
}
```
Analysis:

The program presents the facilities of C++ structures for storing and displaying information of books. In the program, a structure called Book is defined with members such as title, author, subject, and book_id. Besides that two variables, Book1 and Book2 are declared and initialized by two different sets of data. Further the program outputs all the stored values for each book by using cout command. Here we see the use of structures which not only simplify the grouping of related data but also provide easy access to them.

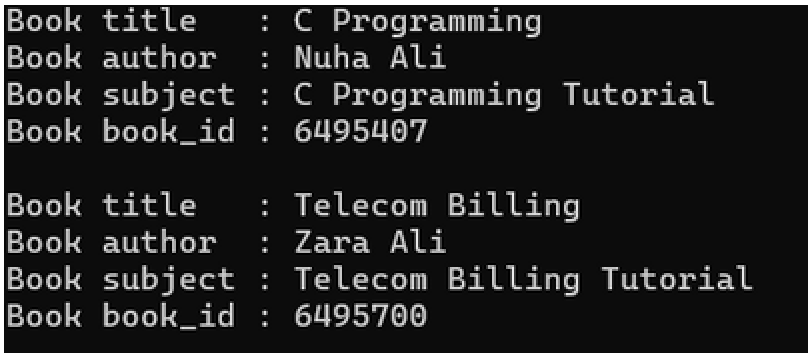
Code:



```
Book 1 title   : C++ Programming
Book 1 author  : Nuha Ali
Book 1 subject : C Programming Tutorial
Book 1 book_id : 6495407

Book 2 title   : Telecom Billing
Book 2 author  : Zara Ali
Book 2 subject : Telecom Billing Tutorial
Book 2 book_id : 6495700
```

```
#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;
}
```

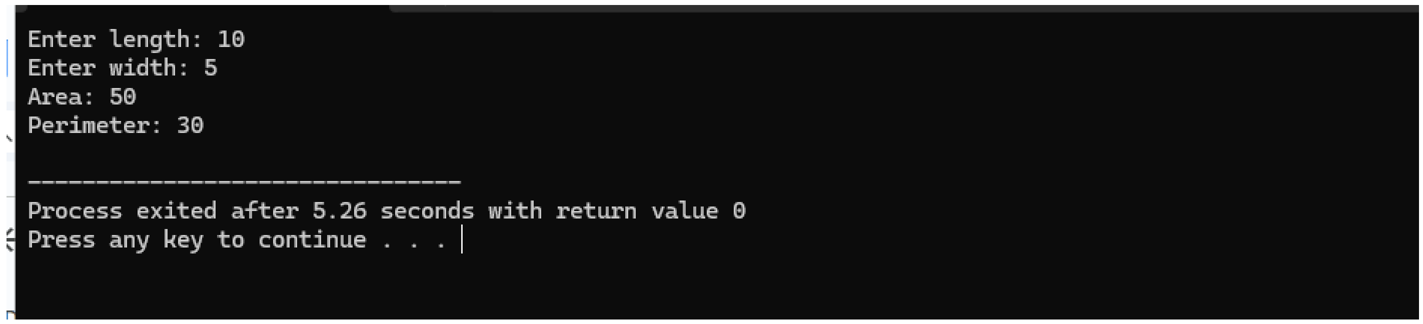
Analysis:
The image code is an example of using structures in C++ along with functions. Firstly, it declares a Book structure that holds the information such as title, author, subject, and book ID. After that, the program initializes two objects of Book, sets their member values, and passes them into the function for printing. In this way, the demonstration of structures passed by value to functions is given, which allows the programmer to simplify the work with multiple related data items. The output is very informative as it shows the complete details of both books, thus the system comprising function and structure is working effectively



```
Book title   : C Programming
Book author  : Nuha Ali
Book subject : C Programming Tutorial
Book book_id : 6495407

Book title   : Telecom Billing
Book author  : Zara Ali
Book subject : Telecom Billing Tutorial
Book book_id : 6495700
```

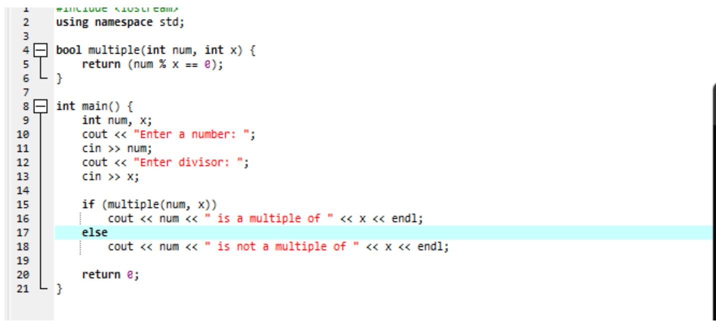
7. Supplementary Activity

```
#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 << "Area: " << area << endl;
   cout << "Perimeter: " << perimeter << endl;
}
```

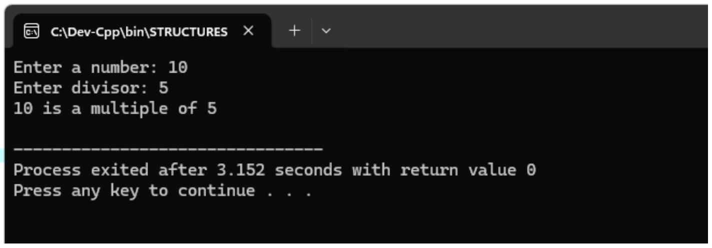
```
int main() {
   Rectangle rect;
   cout << "Enter length: ";
   cin >> rect.length;
   cout << "Enter width: ";
   cin >> rect.width;

   compute(rect);
   return 0;
}
```
Analysis:
In this C++ program, the user enters numbers to find the area and perimeter of a rectangle. At the top, there is a struct named Rectangle. This struct is a blueprint for the main function to use. It holds two numbers (of type double) for the length and width of the rectangle. Next is main which creates a variable of type Rectangle and asks the user to put in the length and width. Another function named compute takes in the rectangle variable, finds the area and perimeter, and prints the final values to the screen. Then main calls compute and gives it the user input.
Code:

```
Enter length: 10
Enter width: 5
Area: 50
Perimeter: 30

--------------------------------
Process exited after 5.26 seconds with return value 0
Press any key to continue . . .
```
```
#include <iostream>
using namespace std;

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

int main() {
   int num, x;
   cout << "Enter a number: ";
   cin >> num;
   cout << "Enter divisor: ";
   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;
}
```
Analysis:
The purpose of this C++ program is to determine whether or not one integer is a multiple of another. To accomplish this, the program includes a boolean function called multiple that takes two integers as arguments.

The multiple function can use the modulo operator (%) to determine whether or not the first number is evenly divisible by the second number. If the remainder when dividing the first number by the second number is zero, the function can return true and otherwise, it can return false. The program's main function is just responsible for prompting the user for a number and a divisor. Finally, the program's boolean function can be called and used inside an if-else statement to print out a message stating whether or not the first number is a multiple of the second.

Code:



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

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

int main() {
    int num, x;
    cout << "Enter a number: ";
    cin >> num;
    cout << "Enter divisor: ";
    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;
}
```



```
Enter a number: 10
Enter divisor: 5
10 is a multiple of 5

--------------------------------
Process exited after 3.152 seconds with return value 0
Press any key to continue . . .
```

8. Conclusion:Overall, this C++ program shows good design. It solves the divisibility problem well. It uses a multiple function. The function's use of the modulo operator is efficient. Its boolean return value fits with the if-else logic in main. The code is easy to read. It demonstrates basic programming principles. It separates tasks using a simple function. The function design is the right choice. The program works well for its job. But it could be better if it checked user input. This would avoid division by 0 and runtime errors. This code is a compact example of logical, well-executed code. It is appropriate for its job. This solution can also be used as an instructional example.