Hands-on Activity 5.2	
Structures	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:9/30/25**
Section:CPE11S1	**Date Submitted:10/2/25**
Name(s): Niel Vincent B. Condino	**Instructor: Engr. Jimlord M. Quejado**

6. Output

```
#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;
}
```

The code begins by including the <iostream> and <string> libraries. It also uses the namespace std to avoid repeatedly writing “std::” . A structure named “Card” is then defined with two members, face and suit, both of type string. Inside the main function, variable “a” and a pointer “aPtr” are declared. It then assigns "Ace" to the face member and "Spades" to the suit member of a. Afterwards, the pointer aPtr is set to point to the memory address of a. The code then shows three different ways of accessing the structure members.Either directly through the variable using the dot operator, through the pointer using the arrow operator, and by dereferencing the pointer and then using the dot operator. In all three cases, the program outputs "Ace of Spades". Finally, the program ends by returning 0.

```
#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;
}
```

The code begins by including two libraries, <iostream> and <string>.It also uses namespace std in order to avoid adding “std::” in the beginning for functions that need it. Then it defines a structure named Books with 4 members named title, author, subject ,and book_id.The first 3 are strings while the last member is an int. Inside the main function, it first declares 2 variables with the type of Books named “Book1” and “Book2”.It then assigns values to the members of “Book1”, giving it a title of "C Programming," an author named "Nuha Ali," a subject of "C Programming Tutorial," and a book ID of “6495407”. Similarly, the members of “Book2” are initialized with different values, it has a title of "Telecom Billing," an author named "Zara Ali," a subject of "Telecom Billing Tutorial," and a book ID of “ 6495700”. The next lines are about printing “Book1” and “Book2” ‘s information. Lastly it returns 0.

```
#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;
}
```

The code begins by including the <iostream> and <string> libraries. The namespace std is used so that standard objects and functions can be called without the std:: prefix. A structure named Books is then defined with four members: title, author, and subject of type string, and book_id of type int. After this, a function named printBook is declared, which takes a Books structure as an argument. Inside the main function, two structure variables, Book1 and Book2, are declared. Their members are assigned values: Book1 is given the title "C Programming", author "Nuha Ali", subject "C Programming Tutorial", and book ID 6495407, while Book2 is assigned the title "Telecom Billing", author "Zara Ali",

subject "Telecom Billing Tutorial", and book ID 6495700. Once initialized, both structures are passed by value to the printBook function, which displays their contents. A blank line (cout << endl;) is used to separate the outputs for readability. The printBook function is then defined after main. It takes a Books object as input and uses cout to print each of its members (title, author, subject, and book ID). As a result, when the code runs, the details of both Book1 and Book2 are printed. Finally, the program ends with return 0;.

7. Supplementary Activity

```cpp
#include <iostream>

using namespace std;

struct rectMeasureInput{
    float length;
    float width;
};

struct rectMeasureOutput{
    float perimeter;
    float area;
};

rectMeasureOutput calcPerAndArea(rectMeasureInput given){
    float l = given.length;
    float w = given.width;

    rectMeasureOutput answers;
    answers.perimeter = 2*(l+w);
    answers.area = l*w;
    return answers;
}

int main(){
    cout << "Rectangle Perimeter and Area Calculator\n";
    rectMeasureInput givens;
    float inputL, inputW;

    cout << "Enter length (cm):";
    cin >> inputL;
    cout << "Enter width (cm):";
    cin >> inputW;

    givens.length = inputL;
    givens.width = inputW;

    rectMeasureOutput outputs = calcPerAndArea(givens);
    float perimeter = outputs.perimeter;
    float area = outputs.area;

    cout << "Calculated perimeter is " << perimeter << " centimeters and calculated area is " << area << " squared centimeters";
    return 0;
}
```

```
Rectangle Perimeter and Area Calculator
Enter length (cm):4
Enter width (cm):5
Calculated perimeter is 18 centimeters and calculated area is 20 squared centimeters
--------------------------------
Process exited after 2.764 seconds with return value 0
Press any key to continue . . .
```

Code Analysis:

The code begins by including the <iostream> library. After that it uses namespace std in order to avoid using “std::” for the functions that need it. Two structures are then defined, rectMeasureInput with members with a type of float and named length and width, and rectMeasureOutput with members that also have a type of float that are named perimeter and area to store the calculated results. A function named calcPerAndArea is defined to take a rectMeasureInput structure as parameter and return a rectMeasureOutput structure. Inside this function, the rectangle’s length and width are extracted by storing the length inside a local variable named l and the width inside the local variable w. Float data types are used in order for it to still have correct results even if the inputted number has decimal values. Then the perimeter is calculated using the formula 2 * (l + w) while the area is calculated using l * w. These values are stored in a rectMeasureOutput structure named “answers” and returned. In the main function, the program first prints a heading message “Rectangle Perimeter and Area Calculator”. It then declares a rectMeasureInput variable named givens and two

float variables for user input. The user is asked to enter the rectangle's length and width in centimeters, which are then stored in givens. The function calcPerAndArea is called with givens as an argument, and its return value is stored in a rectMeasureOutput variable named outputs. The program extracts the perimeter and area from this structure and stores them in separate variables. Finally, it prints out the calculated perimeter and area in a formatted text, before ending with return 0.

```cpp
#include <iostream>

using namespace std;

bool isMultiple(int inputInteger,int inputMultiple){
    if (inputInteger % inputMultiple == 0){
        return true;
    }
    else {
        return false;
    }
}

int main(){
    int input;
    int inputMultipleOf;

    cout << "Input number:";
    cin >> input;

    cout << "Check if is multiple of ";
    cin >> inputMultipleOf;

    if (isMultiple(input,inputMultipleOf)){
        cout << input << " is a multiple of " << inputMultipleOf;
    }
    else {
        cout << input << " is not a multiple of " << inputMultipleOf;
    }

    return 0;
}
```

```
Input number:100
Check if is multiple of 10
100 is a multiple of 10
--------------------------------
Process exited after 6.814 seconds with return value 0
Press any key to continue . . .
```

```
Input number:7
Check if is multiple of 2
7 is not a multiple of 2
--------------------------------
Process exited after 1.499 seconds with return value 0
Press any key to continue . . .
```

The code begins by including the <iostream> library and uses namespace std. It defines a function named isMultiple that takes two integers, inputInteger and inputMultiple, as parameters and returns a data type of bool. Inside the function, it

checks if inputInteger is divisible by inputMultiple using the modulus or % operator. If the remainder is zero, the function returns true, which means inputMultiple is a multiple of inputInteger, otherwise, it returns false. In the main function, two integer variables, input andinputMultipleOf are initialized. The user is first prompted to enter an integer, which is stored in input. Next, the user is asked to provide another integer to check if it is a multiple of input, which is stored in inputMultipleOf. The code then calls the isMultiple function with these two values. If the function returns true, it prints a message stating that the first number is a multiple of the second. If it returns false, it prints a message stating that the first number is not a multiple of the second. Finally, the code ends by returning 0.

8. Conclusion

In the lesson, I learned about creating structures and functions on C++. Structures are interesting on how they work because they are kind of similar to arrays in that they hold data. I think their main difference is in an array it just holds data that have very similar data types. As opposed to a structure, it can hold many data types. Structures feels like a new data type on how they are initialized, they are typed out before the name of the variable. Functions will also be useful as they are repeated blocks of codes. I can just create a single function if I need a block of code to be repeated in my program. They can improve the code's readability a lot and can lessen the number of lines it makes. The code analysis part of the examples showed me how structures and functions work on C++. There are different ways of accessing structures like using the dot operator, the arrow operator on a pointer, and using a dot operator on a dereferenced pointer. The second example also shows how to access a structure member and set a value to them using a dot operator. The last example then showed how I can utilize a function in a way that it sets a value for each member of a structure.
The supplementary activity made me utilize my learnings on the structures and functions. The first one combines both of the concepts. It is pretty simple to implement as they are already similar to an array but can hold different data types. I think it is also better than an array in a way you can name its members but only when the data it is holding is not large where an array will be a better option. The second part of the activity is pretty simple also but uses a pretty niche operator for finding a remainder. This operator is very useful if you want to know if a certain number is a factor or a multiple of a certain number. Overall, I will say that even though the concept of a function was already introduced in my math subjects in the past, applying them on programming so it can optimize my code's readability and creating structures are very helpful for holding different data types inside of just one variable. I also think the concept of functions and structures will be essential in the higher levels of programming.

9. Assessment Rubric