Lab's Scope:

- **Struct and enum**

Problem 1:

Declare a structure to represent a rectangle where a rectangle has members length, width, perimeter, and area. Write a C++ application that declares a variable of a struct rectangle where the height and width are given by the user. The application should then calculate the perimeter and area of the rectangle and store them in member variables perimeter, and area of the struct. The application should print members of the struct area and perimeter.

*Hint: Area of rectangle = length * width.*
Perimeter of a rectangle = 2(length + width)*

Solution:

```
#include <iostream>
using namespace std;
struct rectangle
{
    int length;
    int width;
    int area;
    int perimeter;
};

int main() {
    rectangle r;
    cout << "Enter the length:" << endl;
    cin >> r.length;
    cout << "Enter the width:" << endl;
    cin >> r.width;
    r.area = r.length * r.width;
    r.perimeter = 2 * (r.length + r.width);

    cout << "The area of the rectangle is:" << r.area << endl;
    cout << "The perimeter of the rectangle is:" << r.perimeter << endl;

    return 0;
}
```

Problem 2:

Declare a structure to represent a student where a student has a name and age. Write a C++ application that declares an array of struct students.

- Write a function that gets information about all students from the user.
- Write a function that given the array of students, should print only the students with age more than or equal to 14 years.
- Let the application call the functions.

Solution:

```
#include <iostream>
using namespace std;
struct Student
{
    string name;
    int age;
};
const int SIZE = 3;
//////////////////////////////////////////
void printStudents(Student arr[])
{
    for (int i = 0; i < SIZE; i++)
    {
        if (arr[i].age >= 14)
        {
            cout << "Student name:" << endl;
            cout << arr[i].name << endl;
            cout << "Student age:" << endl;
            cout << arr[i].age << endl;
        }
    }
}
//////////////////////////////////////////
void getsInfo(Student arr[])
{
    for (int i = 0; i < SIZE; i++)
    {
        cout << "Enter student name:" << endl;
        cin >> arr[i].name;
        cout << "Enter student age:" << endl;
        cin >> arr[i].age;
    }
}
//////////////////////////////////////////
int main() {
    Student arr[SIZE];
    getsInfo(arr);
    printStudents(arr);
    return 0;
}
```

Problem 3:

Declare a structure complex to represent a complex number where a complex number has members real (float) and imaginary (float). Write a C++ application that declares two complex numbers of struct complex. Write three functions to add, subtract, and multiply two complex numbers and display the results.

Hint: Adding two complex numbers by adding their real members and storing them in the real member of the result, then adding imaginary members and storing them in the imaginary member of the result. (the same concept for subtraction and multiplication)

Solution:

```
#include <iostream>
using namespace std;
struct complex
{
    float real;
    float imaginary;
};
//////////////////////////////////////////////
void add(complex a, complex b, complex c)
{
    c.real = a.real + b.real;
    c.imaginary = a.imaginary + b.imaginary;
    cout << "Addition result:" << endl;
    cout << "The real part is:" << c.real<<endl;
    cout << "The imaginary part is:" << c.imaginary << endl;
 }
//////////////////////////////////////////
void subtract(complex a, complex b, complex c)
{
    c.real = a.real - b.real;
    c.imaginary = a.imaginary - b.imaginary;
    cout << "Subtraction result:" << endl;
    cout << "The real part is:" << c.real << endl;
    cout << "The imaginary part is:" << c.imaginary << endl;
}
//////////////////////////////////////////////////////////////
void multiply(complex a, complex b, complex c)
{
    c.real = a.real * b.real;
    c.imaginary = a.imaginary * b.imaginary;
    cout << "Multiplication result:" << endl;
    cout << "The real part is:" << c.real << endl;
    cout << "The imaginary part is:" << c.imaginary << endl;
}
//////////////////////////////////////////////////////////////
```

```
int main()
{
    complex a, b, c;
    a.real = 5.5;
    a.imaginary = 3.5;
    b.real = 6.5;
    b.imaginary = 2.5;
    c.real = 0;
    c.imaginary = 0;
    add(a, b, c);
    subtract(a, b, c);
    multiply(a, b, c);
    return 0;
}
```

Problem 4:

Write a structure Customer to store the name, account number, and balance of customers.

- Write a function that lets the customer withdraw from his account an amount of money that affects his balance and print the new balance.
- Write a function that lets the customer deposit to his account an amount of money that affects his balance and print the new balance.
- Create an application that lets a customer withdraw and deposit from his account.

Solution:

```
#include <iostream>
using namespace std;
struct Customer
{
    int account_number;
    string name;
    float balance;
};
//////////////////////////////////////////////////////////
void withdraw(Customer& c, float amount)
{
    if (c.balance >= amount)
        c.balance -= amount;
    cout << "Account number: " << c.account_number << endl;
    cout << "Customer name: " << c.name << endl;
    cout << "balance: " << c.balance << endl;
}
```

```
void deposit(Customer& c, float amount)
{
    c.balance += amount;
    cout << "Account number: " << c.account_number << endl;
    cout << "Customer name: " << c.name << endl;
    cout << "balance: " << c.balance << endl;
}
//////////////////////////////////////////////////
int main()
{
    Customer c;
    c.account_number = 1234;
    c.name = "Ali";
    c.balance = 50000;
    float amount;
    cout << "Enter amount to withdraw:" << endl;
    cin >> amount;

    withdraw(c, amount);
    cout << "Enter amount to deposit:" << endl;
    cin >> amount;
    deposit(c, amount);
    return 0;
}
```

Problem 5:

Write a program that involves a struct called Employee that has members ID, name, salary, and performance. Declare an array of 3 employees.

- Write a function to get the information of each Employee.
- Write a function that increments the salary of the employee by 20% if his performance is more than or equal to 80%.
- Write a function that prints the employees' data after salary change.

Solution:

```
#include <iostream>
using namespace std;
struct Employee
{
    int ID;
    string name;
    float salary;
    float perfomance;
};
const int SIZE = 3;
```

```
Employee getInfo()
{
    Employee e;
    cout << "Enter Employee name:";
    cin >> e.name;
    cout << "Enter Employee ID:";
    cin >> e.ID;
    cout << "Enter Employee salary:";
    cin >> e.salary;
    cout << "Enter Employee performance:";
    cin >> e.perfomance;
    return e;
}
//////////////////////////////////////////////////////
void printEmployees(Employee arr[])
{
    for (int i = 0; i < SIZE; i++)
    {
        cout << "Employee " << i << endl;
        cout << arr[i].name<<endl;
        cout << arr[i].ID<<endl;
        cout << arr[i].salary<<endl;
        cout << arr[i].perfomance<<endl;
        cout << endl;
    }
}
//////////////////////////////////////////////////////////////
 void increase_Salary(Employee arr[])
 {
        for (int i = 0; i < SIZE; i++)
        {
            if (arr[i].perfomance >= 80)
                arr[i].salary += (arr[i].salary * 0.2);
        }
 }
//////////////////////////////////////////////////////////////////
int main()
{
    Employee arr[SIZE];
    for (int i=0; i<SIZE; i++)
    {
        arr[i]=getInfo();
    }
    increase_Salary(arr);
    printEmployees(arr);
    return 0;
}
```

Problem 6:

Write an application that involves an enum representing the days of the week, starting that Sunday as day number 1. The application should declare a variable the enum and take an int value from the user and print out which day of the week it is according to the number entered.

Solution:

```
#include <iostream>
using namespace std;
enum week_day
{
    Sunday = 1,
    Monday,
    Tuesday ,
    Wednesday,
    Thursday,
    Friday,
    Saturday
};

int main()
{
    int day;
    cout << "Enter the week day number: ";
    cin >> day;
    switch (day)
    {
        case Sunday: cout << "Today is Sunday." << endl;
            break;
        case Monday: cout << "Today is Monday." << endl;
            break;
        case Tuesday: cout << "Today is Tuesday." << endl;
            break;
        case Wednesday: cout << "Today is Wednesday." << endl;
            break;
        case Thursday: cout << "Today is Thursday." << endl;
            break;
        case Friday: cout << "Today is Friday." << endl;
            break;
        case Saturday: cout << "Today is Saturday." << endl;
            break;
        default: cout << "Invalid Input." << endl;
            break;
    }
    return 0;
}
```