Objective:

- The tasks given below will help you feel easy with keyword 'struct'.
- You will get to know the basics of struct variable passing by value/reference.
- You will also get to know about the struct variable layout in memory.

Task 1: Focus on defining struct members

A. Write a program that uses a structure named MovieData to store the following information about a movie:

Title
Director
Year Released
Running Time (in minutes)

The program should create two MovieData variables, store values in their members, and pass each one, in turn, to a function that displays the information about the movie in a clearly formatted manner.

B. Corporate Sales Data

Write a program that uses a structure to store the following data on a company division:

Division Name (such as East, West, North, or South)

First-Quarter Sales

Second-Quarter

Sales Third-Quarter Sales

Fourth-Quarter Sales

Total Annual Sales

Average Quarterly Sales

The program should use four variables of this structure. Each variable should represent one of the following corporate divisions: East, West, North, and South. The user should be asked for the four quarters sales figures for each division. Each division's total and average sales should be calculated and stored in the appropriate member of each structure variable. These figures should then be displayed on the screen.

Task 2: Focus on pointers, array manipulation and memory layout of struct members.

A. What will be displayed on console when the following code is executed?

```
struct Time
{
	int hours;
	int minutes;
	int seconds;
};
void main()
{
	Time t;
	t.hours = 10;
	t.minutes = 17;
	t.seconds = 23;
	int * p = (int*) &t;
	p[0] = 20;
	p[1] = 15;
	p[2] = 30;
	cout<<t.hours<<endl<<t.minutes<<endl<<t.seconds<<endl;
}
```

B. Give output of the following code segment.

```
struct Wow
{
	int rollNo;
	char name[48];
	float cgpa;
};

void main()
{
	Wow w = {512, "Ahmed", 3.7};

	cout<< *( (float*) ( (char*)&w + sizeof(w.rollNo) + sizeof(w.name)) );
}
```

Task 3: Storing Rational Numbers
As discussed in lecture: Define all the following functions for the following Rational struct

```
struct Rational
{
        int numerator;
        int denominator;
};
```

Functions to implement:

3.1. void inputRational(Rational &)
This function takes input from user for numerator and denominator and stores it in received Rational variable.

3.2. void printRational(Rational &)
This function prints on screen the received Rational variable.

3.3. Rational addRational(Rational a, Rational b)
This function returns a Rational variable, which is the addition of two received rational variables.

3.4. Rational diffRational(Rational a, Rational b)
This function returns a Rational variable, which is the difference of two received rational variables.

3.5. Rational divRational(Rational a, Rational b)
This function returns a Rational variable, which is the division of two received rational variables.

3.6. void reduce(Rational & a)
This function modifies the received rational variables by reducing the numerator and denominator. For Example, if the received variable has 12/30 then this function change it to 2/5.

Task 4: Storing time information.
Implement the struct 'Time' with the following given members

Members:

```
int hour;
int minute;
int second;
```

4.1. void inputTime(Time & t); // set hour, minute, second
4.2. void printTwentyFourHourFormat(Time); // print universal time
4.3. void printTwelveHourFormat(Time); // print standard time
4.4. Void incSec(Time &, int = 1);// increment in the second of the received time variable
// default increment is 1
4.5. Void incMin(Time &, int = 1); // increment in the minute of the received time variable
// default increment is 1
4.6. Void incHour(Time &, int = 1); // increment in the hour of the received time variable
// default increment is 1
4.7. bool isTimeSame(Time &, Time &); // returns true if two received times are same