The goal of this lab is to help you get familiar with C++ **`struct`**s. It is also an opportunity to practice writing and debugging a program with multiple functions. Be sure that you read this entire document before you begin coding.

Due Date

You must *submit* the source code for the solution to this lab exercise to *Moodle* by

Tuesday, July 15, 2025

in order to receive full credit for this work. You must also *demonstrate* the solution to the instructor during class, at the earliest opportunity.

Programming Exercise

1. Read and understand this entire document before writing any code.
2. Declare a **`struct`** named **`MovieData`** to store the following information about a movie:

Field Name	Data Type	Description
`title`	`string`	Movie Title
`director`	`string`	Movie Director
`yearReleased`	`int`	Year Released
`runningTime`	`double`	Running time in minutes

3. Write a program that uses dynamic memory allocation to create an array of **`MovieData`** structs, populates the array with data, and does some processing with that data. Your program MUST be organized as this document describes:
   The **main** function in your program **must** perform the following steps:
   a. Ask the user to enter how many movies they wish to process. The number specified by the user will be used for the dynamic memory allocation.
   b. Allocate memory for an array of the **`MovieData`** structs, using the size value from the user input.
   c. Call the **populateMovieDataArray** function.
   d. Call the **displayMovieDataArray** function.
   e. Call the **findLongestMovie** function. This function must return a pointer to the array element that contains information for the longest movie. (**NOTE**: this function does ***not*** display the **`MovieData`** details, but only returns a pointer to the correct array element.)
   f. Call the **displayMovie** function, passing it the pointer that was returned by **findLongestMovie**.
   g. Before the program exits, it must **delete** the dynamically allocated array of **MovieData** structs.

Program Design

Your program **must** have functions, as described in the next sections of this document. Start by reviewing the design of the solution for **Lab09a**. The **`main`** function dynamically allocates an array of **`MovieData`** structs, and then calls other functions to perform the various tasks.

The list of **function prototypes** shown below provides guidance about how you must organize your program.

```
void displayMovie(MovieData *moviePtr);
void populateMovieDataArray(MovieData *arrayPtr, int arraySize);
void displayMovieDataArray(MovieData *arrayPtr, int arraySize);
MovieData *findLongestMovie(MovieData *arrayPtr, int arraySize);
```

Function: main

- Ask the user for the desired array size.
- Allocate the memory for the array of structs.
- Call **populateMovieDataArray**.
- Call **displayMovieDataArray**
- Call **findLongestMovie** to obtain a pointer to the struct for the longest movie.
- Call **displayMovie**, using the pointer that was returned by **findLongestMovie** as the argument for the **displayMovie** function.
- De-allocate the memory for the array of structs.

Function: displayMovie

This function has one input parameter:

`moviePtr = address of a` **`MovieData`** `struct`

The **`MovieData`** struct (pointed to by **`moviePtr`**) contains information about **one** movie. This function must output the data fields of the **`MovieData`** struct to **`cout`**, in a format similar to the examples in the **Sample Output** section of this document. **NOTE:** This function is the ONLY code that displays the details of a **`MovieData`** struct. (Any *other* code that wishes to display the details of a **`MovieData`** struct must call this function.)

Function: populateMovieDataArray

This function has two input parameters:

`arrayPtr =` address of beginning of array
`arraySize =` number of elements in the array.

The **populateMovieDataArray** function must contain a loop that prompts the user to enter data values for each field of each struct in the array.

Function: displayMovieDataArray

This function has two input parameters:

`arrayPtr` = address of beginning of array
`arraySize` = number of elements in the array.

The **displayMovieDataArray** function must contain a loop that calls the **displayMovie** function for each element of the array, formatted as follows:

- Display the hexadecimal address of each array element (that is, each struct in the array),
- Call the **displayMovie** function to display the details about the movie.

(Observe the **Sample Output** section of this document as an example of the formatting.)

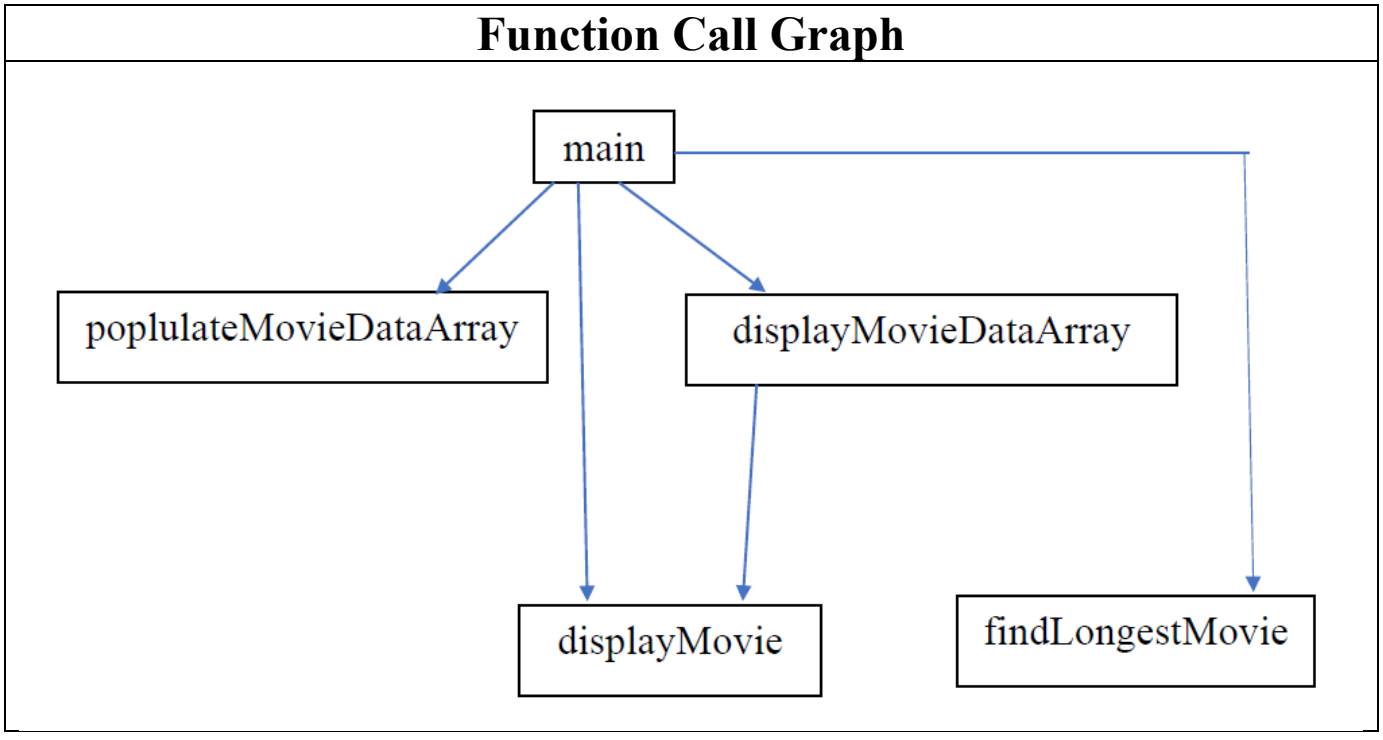
Function: findLongestMovie

This function has two input parameters:

`arrayPtr` = address of beginning of array
`arraySize` = number of elements in the array.

The **findLongestMovie** function must contain a loop that scans the array to identify the array element with the longest **runningTime** value.

The **findLongestMovie** function must return (to the caller) a **pointer** to the array element that contains the data for the longest movie. (This function does NOT call the **displayMovie** function, or display the **`MovieData`** details.)



Sample Input/Output

The table that begins below, and continues on the following page, contains sample input and output for the solution to this lab exercise. In this example, we have indicated which text the user types by showing it in a larger, **bold** font. In actuality, all text would appear in the same size font, with no bold characters.

Sample Input/Output

```
Enter desired array size: 6
arrayPtr = 000001E7D9EF9268
Enter Title 0: The Light Between Oceans
Enter Director 0: Derek Cainfrance
Enter Year Released 0: 2016
Enter running time (minutes) 0: 133
Enter Title 1: News of the World
Enter Director 1: Paul Greengrass
Enter Year Released 1: 2020
Enter running time (minutes) 1: 118
Enter Title 2: 2001: A Space Odyssey
Enter Director 2: Stanley Kubrick
Enter Year Released 2: 1968
Enter running time (minutes) 2: 142
Enter Title 3: The Sound of Music
Enter Director 3: Robert Wise
Enter Year Released 3: 1965
Enter running time (minutes) 3: 174
Enter Title 4: Finding Nemo
Enter Director 4: Andrew Stanton
Enter Year Released 4: 2002
Enter running time (minutes) 4: 100
Enter Title 5: Hidden Figures
Enter Director 5: Ted Melfi
Enter Year Released 5: 2016
Enter running time (minutes) 5: 127
000001E7D9EF9268:  arrayPtr[0] =
        Title       : The Light Between Oceans
        Director    : Derek Cainfrance
        Released    : 2016
        Running Time: 133 minutes

000001E7D9EF92C8:  arrayPtr[1] =
        Title       : News of the World
        Director    : Paul Greengrass
```

Sample Input/Output

```
        Released    : 2020
        Running Time: 118 minutes

000001E7D9EF9328:  arrayPtr[2] =
        Title       : 2001: A Space Odyssey
        Director    : Stanley Kubrick
        Released    : 1968
        Running Time: 142 minutes

000001E7D9EF9388:  arrayPtr[3] =
        Title       : The Sound of Music
        Director    : Robert Wise
        Released    : 1965
        Running Time: 174 minutes

000001E7D9EF93E8:  arrayPtr[4] =
        Title       : Finding Nemo
        Director    : Andrew Stanton
        Released    : 2002
        Running Time: 100 minutes

000001E7D9EF9448:  arrayPtr[5] =
        Title       : Hidden Figures
        Director    : Ted Melfi
        Released    : 2016
        Running Time: 127 minutes

Longest Movie in list:

        Title       : The Sound of Music
        Director    : Robert Wise
        Released    : 1965
        Running Time: 174 minutes

Longest Movie is:  174 minutes long
DELETING array at arrayPtr = 000001E7D9EF9268
Exit the program.
```

Submit and Demonstrate the Working Program

- Submit the source code file (**`*.cpp`**) for the working program to the *Moodle* assignment for this **Lab Exercise**.
- Demonstrate the working program to the instructor during class.
- Be sure to save a copy of the source file (**`*.cpp`**) in a safe place for future reference.