* Movie Vector(20 pts)
* Employee Exercise (35 pts)
  + Employee Class File(15 pts) – Employee.h
  + Employee Driver Program (15 pts)

**Movie Vector**

Write a program that allows the user to enter movie information, which is then stored in a vector. The user should enter data for at least one movie, but the program should ask if they have additional information to enter. When the user has indicated that they are done entering movie data, the program should output the information stored in the vector

You should use:

1. A struct called Movie to store the following information:

* Title
* Director
* Year Released
* Running Time (in minutes)

1. A vector of type Movie to store the information
2. A function that allows you to obtain the data. The function should have a reference to a movie variable as a parameter, but does not return a value. (Tip: as in the class example the variable should receive all the data before it is inserted into the vector. Insert enough data for at least two movies for this exercise.
3. A function to output the movie data that was entered. This function should accept two parameters: a vector and an integer (for the size of the vector).

Tip:        In case you’re wondering, the header for the function that uses a

vector should have a typical vector declaration within the parentheses.

         Recall: To declare a vector, include the vector template and use the general

syntax:         vector <datatype> name

Sample output is shown below:

**Enter the title of the movie:**The Dark Knight

**Enter the director's name:**Christopher Nolan

**Enter the year the movie was created:**2008

**Enter the movie length (in minutes):**152

**Do you have more movie info to enter?**

**Enter y/Y for yes or n/N for no:**y

**Enter the title of the movie:**The Avengers

**Enter the director's name:**Joss Whedon

**Enter the year the movie was created:**2012

**Enter the movie length (in minutes):**143

**Do you have more movie info to enter?**

**Enter y/Y for yes or n/N for no:**Y

**Enter the title of the movie:**Deadpool

**Enter the director's name:**Tim Miller

**Enter the year the movie was created:**2016

**Enter the movie length (in minutes):**108

**Do you have more movie info to enter?**

**Enter y/Y for yes or n/N for no:**n

**Here is the info that you entered:**

**Movie Title: The Dark Knight**

**Movie Director: Christopher Nolan**

**Movie Year: 2008**

**Movie Length: 152 minutes**

**Movie Title: The Avengers**

**Movie Director: Joss Whedon**

**Movie Year: 2012**

**Movie Length: 143 minutes**

**Movie Title: Deadpool**

**Movie Director: Tim Miller**

**Movie Year: 2016**

**Movie Length: 108 minutes**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Employee Exercise**

For this exercise, you will create two files – the Employee.h file, which will hold the class and member function definitions, and the empDriver.cpp file, which will be used to implement the Employee objects.

Employee.h

Create a class called Employee that includes the following:

1. Two data members:
   1. full name
   2. monthly salary
2. Two constructor that initializes the two data members.
   1. The first constructor should be a default constructor. It takes no parameters, but initializes the name and salary to “ ” (space) and 0 respectively.
   2. The second constructor should take two parameters, and should initialize the name and salary using the values in the parameters
3. Accessor and Mutator functions for each member.

Tip: create inline functions for this exercise.

empDriver.cpp

Create a driver program that tests the Employee class.

1. Create two Employee objects. (You may “hard code” one object to test your 2-paramter constructor. The other object should rely on your default constructor)
2. Prompt the user for input for the object that was not “hardcoded”. Transfer the values to the private class members by using the mutator functions.
3. Use the accessor functions to retrieve the name and salaries for the employees and calculate the yearly salary. Display the name and yearly salary for the objects.
4. Give your employees a 10% raise! Use the mutator functions to update the value of their monthly salary.
5. Repeat step 3 to show that the raise has taken effect.

Sample output is shown below:

**Enter an employee name:**Peter Peterson

**Enter the employee's monthly salary:**4000

**Here are the employees and their yearly salaries**

**Nicole Watterson: $42000**

**Peter Peterson: $48000**

**Here are the employees and their yearly salaries with a 10% raise**

**Nicole Watterson: $46200**

**Peter Peterson: $52800**