ISTANBUL TECHNICAL UNIVERSITY
FACULTY OF COMPUTER AND INFORMATICS ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

BLG102E - INTRODUCTION TO SCIENTIFIC AND ENGINEERING COMPUTATION
2019 -2020 Spring
FINAL EXAM

Release Date:	05.07.2020
Duration:	**45 minutes**
Instructors:	Assoc. Prof. Cüneyd Tantuğ, Assist. Prof. Ayşe Tosun, Turgut Uyar

2 (60 pts). Download the source code and the test code given for this question, add your implementations into this source code, and **submit the modified source code back to Ninova.**

DO NOT change the messages written in printf functions, or in the test code. DO NOT change the main program, ADD the functions and other definitions according to the requirements below.

Please check the sample test code and run your code against the test code in Calico. Your solutions will be assessed with different test cases.

Assume we keep an array (name: movies) of pointers to structs. The array has a fixed size of 50, and it is assumed that the user will not try to add more than 50 movies to this array. The struct's name is **Movie**. In this **Movie** struct, there are three attributes: The title of the movie, how many times it was purchased, and an average rating for the movie. The array initially has two elements as shown in the source code.

2.a. Write a function called **get_purchases** that **searches** a movie in the array using its **title** given by the user. The title search must be case sensitive. If it finds the movie in the array, it returns **the number of times the movie was purchased**. If it does not find the movie in the array, it returns **-1.**

2.b. Write a function called **compute_rating** that **re-calculates the average rating** of a movie based on the **new rating** given by the user. The function first searches for the movie with its title given by the user. If the movie is found, the function adds the new rating to the movie's total rating, and re-calculates the average rating of the movie. The function then updates the rating of the movie. It also increments the number of purchases for the movie by one.
If the movie is not found, we offer the user to add this new movie with the given rating. This part is described in 2.c.

2.c. Write a function called **add_movie** that **adds a new movie** to the array with its **title** and **rating** given by the user. It also sets the number of purchases to 1.

A sample program execution is provided below. Please also check the test code for alternative scenarios.

Program: "Welcome to the movie database. Enter your request (1 for getting the purchases of a movie, 2 for updating the rating of a movie, 3 for exit):"
User: 2
Program: "Enter the movie title:"
User: Harry Potter
Program: "Enter the movie rating (out of 5):"
User: 4
Program: "The movie Harry Potter could not be found in the database. Would you like to add this movie (Y/N)?"
User: Y
Program: "The movie Harry Potter is added to the database with a rating of 4."
Program: "Enter your request (1 for getting the purchases of a movie, 2 for updating the rating of a movie, 3 for exit):"
User: 2
Program: "Enter the movie title:"
User: Harry Potter
Program: "Enter the movie rating (out of 5):"
User: 3
Program: "The movie Harry Potter is updated. The number of purchases is 2. The rating is updated to 3.50."
Program: "Enter your request (1 for getting the purchases of a movie, 2 for updating the rating of a movie, 3 for exit):"
User: 1
Program: "Enter the movie title:"
User: Hellboy
Program: "The movie Hellboy could not be found in the database."
Program: "Enter your request (1 for getting the purchases of a movie, 2 for updating the rating of a movie, 3 for exit):"
User: 1
Program: "Enter the movie title:"
User: Harry Potter
Program: "The movie Harry Potter has been purchased 2 times."
Program: "Enter your request (1 for getting the purchases of a movie, 2 for updating the rating of a movie, 3 for exit):"
User: 3