Final Exercise 2022

There are three programming assignments in this exercise and one set of questions you must answer. Please follow the naming convention specified for each part of the exercise when turning in your assignment.

Note that you must document your programs to receive full credit for the programming parts of this exercise. Include in your program some source-level comments that describe what the program is doing. This does not mean that every line of code must be documented, but the comments should cover the important parts of the program.

FinalExercise_1 – (50 points)

Write a program that requests three integer values from the user. Let's call these values *a, b,* and *c*. The program then calculates *d* using the following formula:

$$d = a^3 + b^2 + c$$

The program should print the calculated value of *d* once the three integers are input.

The program ***must*** validate the user's input. If the user does not enter a valid integer, the program must print a message for the user and exit without doing any calculations. If the user hits enter without entering a value, the program must also exit without doing any calculations.

When complete, turn in your program using the filename FinalExercise_1.c.

FinalExercise_2 – (50 points)

Write a program that reads standard input as a stream of characters until the program encounters the End-of-File or EOF. The program must count the number of spaces and the number of vowels and then report the results. The vowel counts must include the total number of vowels and the count for each vowel: a, e, i, o, and u.

Test the program by piping the contents of the provided FinalExercise_2.txt file as input into your program.

When complete, turn in your source code file using the filename FinalExercise_2.c.

FinalExercise_3 – (100 points)

Write a program that reads employee records from a binary file and places each record into a sorted linked list. The list must be sorted by the employee ID field, which is an integer value in the data. Once all the records have been read, provide the user with the following menu:

1) Display Employee
2) Edit Employee Salary
3) Remove Employee
4) List All Employees

These functions are defined below.

Display Employee

The Display Employee menu choice prompts the user for an employee's ID, finds the employee's record in the list, and prints all the fields in the record. The program should print a message indicating the employee id was not found if the user enters an ID of an employee that is not in the list.

Edit Employee Salary

The Edit Employee Salary menu choice prompts the user for an employee's ID, finds the employee's record in the list, and then prompts the user for a new salary value. The new salary value must be reflected in the data when the list is printed with List All Employees or if the Display Employee function is used for the employee. The program should print a message indicating the employee id was not found if the user enters an ID of an employee that is not in the list.

Remove Employee

The Remove Employee menu choice prompts the user for an employee's ID, finds the employee's record in the list, and then removes the employee record from the list of employees.

List All Employees

The List All Employees menu choice displays all the records in the employee list.

The employee record structure is defined as follows:

```
#define  MAX_NAME              128
#define  MAX_DATE              64
#define  MAX_EMAIL_ADDRESS     128

typedef struct __attribute__((packed, aligned(1))) employee_data {
    int employeeId;
    char firstName[MAX_NAME];
    char lastName[MAX_NAME];
    char middleName[MAX_NAME];
    char emailAddress[MAX_EMAIL_ADDRESS];
    char title[MAX_EMAIL_ADDRESS];
    float annualSalary;
    char dataOfHire[MAX_DATE];
    struct employee_data *pNext;
} EmployeeInformation;
```

The provided file, EmployeeRecords.bin, contains one entry for each employee record. Use the fread() function to read each record and place the record into the linked list. **IMPORTANT: the records in the file do not include a pNext field, therefore the size of each record in the file is:**

```
sizeof(EmployeeInformation) – sizeof(struct employee_data *)
```

where `EmployeeInformation` is a packed structure. This will be important when reading data from the file.

Note that you may use either an array or the `malloc()` function to create new nodes for the list.

When complete, turn in your source code file using the filename FinalExercise_3.c.

FinalExercise_4 - (100 points)

Find the provided FinalExercise_4.txt file. The file contains a list of questions to answer. Edit the FinalExercise_4.txt file by adding your answer to each question.

When complete, turn in your source code file using the filename FinalExercise_4.txt.