COSC220 Fall 2020 – Project # 3   Due: 1/25/2021

PROJECT SPECIFICATIONS:

Develop a mini-database system to manage employee information. Your database should allow users to insert, delete, modify and retrieve employee records. Initially all the data in the database is stored in an external file called "employee.db". When your database program starts, the data in the file is loaded into a dynamically created array of employee first. Then a user interacts with the array during your program execution. When the user decides to exit the program, the data in the array (with all the current updates) is written into the external file (overwrite existing content).

Here are some information for completing this project.

1. Employee information should include first name, last name, SSN, gender, birth date, Salary.

   You should define a class according to below:

```
class Employee{
                string firstName, lastName;
                 string SSN;
                 char gender;
                 float salary;
        public:
            // define contructors
            Employee();
            Employee(string fn, string ln, string ssn, char g, float s);
             // define getter/setter for firstName, lastName, gender, salary
             string getFirstName();
             string getLastName();
             string getSSN();
             char getGender();
             float getSalary();
             void setFirstName(string fn);
             void setLastName(string fn);
             void setGender(char g);
             void setSalary(float sal);
 }
```

2. Load/Store employee information. You should read database information from an external file and insert them into an array of employees at the beginning of your program. Before you end your program, you should save information in the array to the same external file (overwrite its current content). The external file is called employee.db and its format should look like (each field is separated by a while space):

```
Sophie Wang 111-00-1111 F 07-23-1965 100000
David Parker 100-00-1111 M 12-23-1945 200000
John Doe 888-88-888 M 11-02-2004 50000
```

   You should implement the following functions to load/store employee information using external file.

```
/* read database information from a data file and insert each employee information into an
    array and return the pointer to the first element of the array. If the file is empty,
    return NULL
*/
Employee* readDatabase(ifstream* databaseFile);

/* write the information in the array into an external data file.
*/
void writeDatabase(ofstream* databaseFile, Employee* employeeList);

/* write the information in an array into console.
*/
void printDatabase(Employee* employeeList);
```

3. For INSERT operation, users need to provide all the information for an employee. Some information should be validated before they are entered into database. SSN should be exactly nine digits with “-“ in between. First name and last name should contain less than 10 characters each. Gender should be ‘M’, or ‘F’. Birthday must be in the format of MM-DD-YYYY. Salary must be positive. Please note that every employee should have their unique ssn. If a new employee to be added to the database has the same ssn as an existing one, the new employee information will not be entered into the database and a warning message should be given.

   You should use the following prototype for your insert operation. The return value of the function is true or false indicating whether the insertion is successful. The newly inserted element will be the first element.

   ```
   /* Insert a new employee information stored in newEmployee to the array, return NULL
       if insertion fails.
   */
   void insertEmployee(Employee* &employeeList, Employee newEmployee);
   ```

4. For MODIFY operation, users need to provide the SSN first. Then the information for that employee will be retrieved and displayed. The users can modify the all the fields of an employee except the SSN. New information to be inserted must be validated before the employee record is updated. Your program should prompt the users for confirmation for the final updated.

   You should use the following prototype for your modify operation. The return value of the function is a pointer to the newly updated employee and NULL if update fails.

   ```
   /* Find employee with ssn, update its information with those  in newEmployee
   */
   void modifyEmployee(Employee * employeeList, string ssn, Employee newEmployee);
   ```

5. For DELETE operation, users need to provide the SSN. Before the employee record is deleted, your program should prompt the users for confirmation.

You should use the following prototype for your delete operation. The return value of the function is TRUE or FALSE depending if deletion successes or fails.

```
/* Find employee with ssn, delete it from the linked list
*/
bool deleteEmployee(Employee* &employeeList, string ssn);
```

6. For RETRIEVE operation, the database should allow users to retrieve information by any one of the following criteria: 1) SSN 2) last name 3) first name 4) gender 5) salary and display the information about the employees who meet the search criteria. The information displayed should look similar to below (nicely line-up and formatted).

```
                         Employee Information
--------------------------------------------------------------------------------
 First    Last          SSN      Gender     BOD            SALARY
Name     Name
--------------------------------------------------------------------------------
Sophie   Wang      111-00-1111     F      07-23-1965       100000
David    Parker    100-00-1111     M      12-23-1945      200000
John     Doe      888-88-888      M      11-02-2004      50000
```

You should use the following prototype for your retrieve operation.

```
/* Find employee using given search criteria, display the result
*/
void retrieveEmployee(Employee* employeeList, string searchCriteria, string searchStr, float
searchVal);
```

7. You may implement other functions needed during your implementation. For example, function to validate user input, etc.

<u>OTHER REQUIREMENTS</u>

1. Make sure to start work on your project early and make steady progress.
2. Make sure to test your program thoroughly before you hand in your program.
3. Make sure that you give proper names to your variables, program files.
4. Make sure that your program is nicely indented and has meaningful comments.

<u>WHAT TO TURN IN</u>

Email the instructor your source code, testing plan and all the output generated from your testing. We will test your program in the lab on the due date.