COSC220 Spring 2021 – Project # 1.5 Due: 3/7/2021

PROJECT SPECIFICATIONS:

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

However, there are some significant features in this new version:

- You need to use sorted singly linked list with template to store the employee information in your program. You can use re-use the code you write in Lab 4 for this project. You need the templated Node class and all the functions for sorted linked list manipulations.
- employeeSorted.db file stores the employee information in the order of their ssn. When you read in the employee information, you will store them in the order of their ssn as well. When you insert/delete employees, you need to make sure the updated list is still in order.
- You need to use the free function approach to overload operators ">>" or "<<" for the Employee class so you can read in and display an employee information as below (instead of output each piece of information separately):

```
Employee emp;
inputFile >> emp;
```

or

```
outputFile << emp;
```

- You also need to use the member function approach to overload operator "<", ">" and "==" for Employee class so that when you compare two Employee objects using those operators, you are comparing their ssn numbers. So you can write the code below without having :

```
Employee emp1, emp2;
if (emp1 > emp2) // comparing the ssn of emp1 with emp2
```

or

```
if (emp1 < emp2)
```

or

```
if (emp1 == emp2)
```

- Modify the implementation of the function below:
  `void printDatabase(Node<Employee>* eList, int eCount);`
  The new version of the function will allow users to choose to display the whole database in various ways as below. You should use one of the sorting algorithms discussed in class (bubble sort or selection sort).
  - Default: display by the order of their ssn (ascending)

- Display by the order of their last name (ascending) and then first name (ascending) if they share the same last name.
- Display by the order of their birthday (ascending)
- Display by the order of their income (ascending)

Below are the revised prototypes of all the functions you have to implement. Please note that instead of a pointer to an array of Employee, it is a pointer to the first Node object in the linked list and the value field of the Node object is

```
void readDatabase(Node<Employee>* &eList, int & eCount);
void writeDatabase(Node<Employee>* eList, int  eCount);
void printDatabase(Node<Employee>* eList, int  eCount);
bool insertEmployee(Node<Employee>* &eList, int &eCount);
bool modifyEmployee(Node<Employee> * eList, int eCount);
bool deleteEmployee(Node<Employee>* &eList, int &eCount);
void retrieveEmployee(Node<Employee>* eList, int eCount);
```

OTHER REQUIREMENTS

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.

WHAT TO TURN IN

Upload your source code (main.cpp, node.h, node.cpp, employee.h, employee.cpp, auxilary.h, auxilary.cpp) to myClass, put your testing plan and all the output generated from your testing in a pdf file and upload it to myClass as well. The instructor will test your program in the Linux lab during grading. So make sure to develop your project in the Linux environment.