# CSCI 201 – COMPUTER SCIENCE I
# Project 8 (Final Project)
# Payroll System – Version 3
Due Date: May 1, Friday

## Objectives

To provide students with the experience of using classes, arrays, functions and sorting algorithms to solve a problem.

## Problems

Modify Project 6 (Payroll System) so that an employee's data can be added, deleted, modified, sorted and displayed. You are asked to use a menu to provide these services for the users. The menu might look like this:

```
                          Menu
1. Create employee database.
2. Add new employees to the database.
3. Modify employee data.
4. Delete employees from the database.
5. Display employee data to the screen.
6. Sort and display employee data to the screen
7. Quit the system
```

## Requirements

1. You are asked to design two classes to solve this problem. One class, called **Employee**, stores employee's data; the other class, called **EmployeeList**, holds a list of employee objects (called a container).
2. The class **Employee** should have the data members of id, first name, last name, rate, hours and wage. The following member functions should be provided:
   - a constructor to initialize employee's data with first name, last name id, rate and hours
   - A member function that sets the last name
   - A member function that sets the first name
   - A member function that sets the id
   - A member function that sets the rate
   - A member function that sets the hours
   - A member function that retrieves the last name
   - A member function that retrieves the first name
   - A member function that retrieves the id
   - A member function that retrieves the rate
   - A member function that retrieves the hours

- A member function that computes and retrieves the wage.

The class should also have the following overloaded operators:

== This operator should check to see if two employee objects are equal.

!= This operator should check to see if two employee objects are not equal.

>> : This operator should read the employee's data from the keyboard or an input file stream in the order of id, first name, last name, rate and hours.

<<: This operator cause the employee data to be displayed to the screen or an output file stream in the order of id, first name, last name, rate, hours and wage.

3. The class **EmployeeList** should have the following data members:
   - An array that holds a list of employee objects.
   - an integer that stores the number of employee objects in the list.

   The class should have the following member functions and operations:
   - a default constructor.
   - a constructor that initializes the list (container) with employee data from a file. The file name, which should be declared as a C-string, should be passed to the constructor.
   - a member function that adds an employee object to the list. The function should have the employee object as a parameter. It should check if the list has more space to hold an additional employee and report a message properly. The function should forbidden a same object to be added to the database.
   - a member function that deletes an employee data from the list. The function should have an employee object as a parameter. It should check if the list is empty. If so, it reports an error message. Otherwise it checks to see this employee is in the list ( using operators == & != on Employee objects). If not, it reports a message again.
   - a member function that searches for an employee in the list according to the id provided as a parameter and returns the index of the employee if it is in the list. Otherwise it returns a negative value to indicate the employee is not in the list.
   - a member function that sorts the list according to the id.
   - a member function that sorts the list according to the employee's last name.
   - a member function that modifies an employee's data according to the location ( index) in the list. The function should have the index as a parameter. It should ask the user which data member to be modified. For example, the output of this function might look like this:

```
Change id? y/n -->y
Enter new ID: 10001
ID has changed.

Change first name? y/n --> y
Enter new first name: John
```

```
first name has changed.

Change last name? y/n -->n

Change rate? y/n --> y
Enter new rate: 24.5
rate has changed.

Change hours worked? y/n --> y
Enter new hours: 93
hours has changed.
```

- a member function that returns the total number of employees in the list.
- a member function that returns an employee object according to the index provided as a parameter to the function.

The class should also have the following overloaded operators

>> : This operator should read a list of employee objects from the keyboard or an input file stream to the list.

<<: This operator cause each employee in the list to be displayed to the screen or an output file stream.

4. The application program should provide a menu as follows:

    1. Create employee database.
    2. Add new employees to the database.
    3. Modify employee data
    4. Delete employees from the database.
    5. Display employee data to the screen.
    6. Sort and display employee data to the screen
    7. Quit the system

    For each task in the menu, use a function to solve the problem. You might need to call some member functions of EmployeeList in the function implementation. These functions can be declared as follows:

```
void inputEmployeeData();
void addNewEmployees();
void modifyEmployeeData();
void deleteEmployees();
void displayEmployeeData();
void sortAndDisplayEmployeeData();
```

    [Hint] The function `modifyEmployeeData` deletes employee data by ids. So it should ask the user to enter employee id first and search for the employee in the list. Once the

employee location in the list has been found, it would be easy to modify employee's data by calling modify member function in the EmployeeList. See the output sample of this function from the following test run.

## An Example of Test Run

The output of your program might look like this:

```
csci>a.out
*****************************************************
**                        Menu                     **
** 1. Create employee database.                    **
** 2. Add new employees to the database.           **
** 3. Modify employee data.                        **
** 4. Delete employees from the database.          **
** 5. Display employee data to the screen.         **
** 6. Sort and display employee data to the screen **
** 7. Quit the system                              **
*****************************************************

Please enter your choice: 1
Enter an employee's data by the order of ID number, first name, last
name, rate, hours:
123450 John Smith 12.5 80

Another employee? y/n --> y
Enter an employee's data by the order of ID number, first name, last
name, rate, hours:
123123 Bob Johnson 20 98
Another employee? y/n --> y
Enter an employee's data by the order of ID number, first name, last
name, rate, hours:
125689 Mary Anderson 10.5 20

Another employee? y/n --> y
Enter an employee's data by the order of ID number, first name, last
name, rate, hours:
100251 Jane Carter 45 40

Another employee? y/n --> n
The employee database has been created and saved in
"EmployeeDatabse.dat"
```

```
**********************************************************
**                        Menu                          **
** 1. Create employee database.                         **
** 2. Add new employees to the database.                **
** 3. Modify employee data.                             **
** 4. Delete employees from the database.               **
** 5. Display employee data to the screen.              **
** 6. Sort and display employee data to the screen      **
** 7. Quit the system                                   **
**********************************************************

Please enter your choice: 2
Enter new employee's data by the order of ID number, first Name, last
Name, rate, hours:
108790 Jim Carry 8.5 40

Another one? Y
Enter new employee's data by the order of ID number, first Name, last
Name, rate, hours:
108790 Jim Carry 8.5 40

This employee has already been in the database.

Another one? n

See updated database in the file "EmployeeDatabase.dat".
```

```
**********************************************************
**                        Menu                          **
** 1. Create employee database.                         **
** 2. Add new employees to the database.                **
** 3. Modify employee data.                             **
** 4. Delete employees from the database.               **
** 5. Display employee data to the screen.              **
** 6. Sort and display employee data to the screen      **
** 7. Quit the system                                   **
**********************************************************
Please enter your choice: 5
```

```
XXX Company Payroll System:

        ID     First Name      Last Name      Rate     Hours      Wage
    123450           John          Smith      12.5        80      1000
    123123            Bob        Johnson        20        98      1960
    125689           Mary       Anderson      10.5        20       210
    100251           Jane         Carter        45        40      1800
    108790            Jim          Carry       8.5        40       340
```

```
**********************************************************
**                       Menu                           **
** 1. Create employee database.                         **
** 2. Add new employees to the database.                **
** 3. Modify employee data.                             **
** 4. Delete employees from the database.               **
** 5. Display employee data to the screen.              **
** 6. Sort and display employee data to the screen      **
** 7. Quit the system                                   **
**********************************************************
Please enter your choice: 6


1. Sort by id.
2. Sort by last name.
Enter your choice: 1
```

```
XXX Company Payroll System:
        ID     First Name      Last Name      Rate     Hours      Wage
    100251           Jane         Carter        45        40      1800
    108790            Jim          Carry       8.5        40       340
    123123            Bob        Johnson        20        98      1960
    123450           John          Smith      12.5        80      1000
    125689           Mary       Anderson      10.5        20       210
```

```
*****************************************************
**                      Menu                       **
** 1. Create employee database.                    **
** 2. Add new employees to the database.           **
** 3. Modify employee data.                        **
** 4. Delete employees from the database.          **
** 5. Display employee data to the screen.         **
** 6. Sort and display employee data to the screen **
** 7. Quit the system                              **
*****************************************************
Please enter your choice: 6

1. Sort by id.
2. Sort by last name.
Enter your choice: 2
```

---

```
XXX Company Payroll System:

        ID     First Name      Last Name      Rate      Hours       Wage
    125689           Mary       Anderson      10.5         20        210
    108790            Jim          Carry       8.5         40        340
    100251           Jane         Carter        45         40       1800
    123123            Bob        Johnson        20         98       1960
    123450           John          Smith      12.5         80       1000
```

---

```
*****************************************************
**                      Menu                       **
** 1. Create employee database.                    **
** 2. Add new employees to the database.           **
** 3. Modify employee data.                        **
** 4. Delete employees from the database.          **
** 5. Display employee data to the screen.         **
** 6. Sort and display employee data to the screen **
** 7. Quit the system                              **
*****************************************************
Please enter your choice: 3

Enter employee's current id: 110021
No such worker in the database.

Modify another employee? y/n → y
Enter employee's current id: 123123
This employee has following data:
    123123            Bob        Johnson        20         98       1960
```

```
Change id? y/n --> n
Change first name? y/n --> n
Change last name? y/n --> y
Enter new last name: Bush
The last name has been changed.

Change rate? y/n --> y
Enter new rate: 35
The rate has been changed.

Change hours worked? y/n --> n

Modify another employee? y/n → n
See the modified result in "EmployeeDatabase.dat".

*****************************************************
**                      Menu                       **
** 1. Create employee database.                    **
** 2. Add new employees to the database.           **
** 3. Modify employee data.                        **
** 4. Delete employees from the database.          **
** 5. Display employee data to the screen.         **
** 6. Sort and display employee data to the screen **
** 7. Quit the system                              **
*****************************************************
Please enter your choice: 4

Enter the employee's data to delete by the order of id, first Name,
last Name, rate, hours:
125689 Mary Anderson 10.5 20
This employee's data is deleted.

Another employee to delete? y/n → y
Enter the employee's data to delete by the order of id, first Name,
last Name, rate, hours:
124100 Mary Carter 7.5 20
No such employee.

Another employee to delete? y/n → n
```

```
**************************************************
**                      Menu                    **
** 1. Create employee database.                 **
** 2. Add new employees to the database.        **
** 3. Modify employee data.                     **
** 4. Delete employees from the database.       **
** 5. Display employee data to the screen.      **
** 6. Sort and display employee data to the screen  **
** 7. Quit the system                           **
**************************************************
Please enter your choice: 7

Thanks for using XXXX Company Payroll System!
```

## Other Requirements

- Write a test program for each class individually before you start writing the application program. The program should test each member function in the class.
- Provide a good documentation in each class header file. The following example demonstrates how to provide such a documentation for a class.

```
// FILE: point.h
// CLASS PROVIDED: point
//
// CONSTRUCTOR for the point class:
//   point(double initial_x = 0.0, double initial_y = 0.0)
//     Postcondition: The point has been set to (initial_x, initial_y).
//
// MODIFICATION MEMBER FUNCTIONS for the point class:
//   void shift(double x_amount, double y_amount)
//     Postcondition: The point has been moved by x_amount along the x
//        axis and by y_amount along the y axis.
//
//   void rotate90( )
//     Postcondition: The point has been rotated clockwise 90 degrees
//         around the origin.
//
// CONSTANT MEMBER FUNCTIONS for the point class:
//   double get_x( ) const
//     Postcondition: The value returned is the x coordinate of the
//         point.
//
//   double get_y( ) const
//     Postcondition: The value returned is the y coordinate of the
//         point.
//
// VALUE SEMANTICS for the point class:
//    Assignments and the copy constructor may be used with point
//        objects.
```

```
#ifndef MAIN_SAVITCH_POINT_H
#define MAIN_SAVITCH_POINT_H

 class point
 {
    public:
        // CONSTRUCTOR
        point(double initial_x = 0.0, double initial_y = 0.0);
        // MODIFICATION MEMBER FUNCTIONS
        void shift(double x_amount, double y_amount);
        void rotate90( );
        // CONSTANT MEMBER FUNCTIONS
        double get_x( ) const { return x; }
        double get_y( ) const { return y; }
    private:
        double x; // x coordinate of this point
        double y; // y coordinate of this point
   };

#endif
```

## What to turn in

- The following source files:
  - Employee.h & Employee.cpp
  - EmployeeList.h & EmployeeList.cpp
  - Application program
- The script file from several test runs by your test data.
- A user document (please see "**Requirements for Programming Project**" posted on D2L course page).