

**EE 2372 Software Design I**  
**Assignment #6: Employee Record Management System**  
**Due Date: End of Week 15 (before Midnight) [May 11, 2014]**

**Objectives:**

- 1) To learn to utilize data structures, including linked lists.
- 2) To sharpen your skills using decision statements, iterative control statements, arrays, pointers, structures and functions.
- 3) To learn to use GNU make to automate building your software.

**Tasks:**

- 1) You will create a program that manages employee records.
- 2) Create a header file (lastname\_employeeec.h) that defines an employee data structure (sEMPLOYEE) that can be linked onto a linked list. The data structure should have the following fields:
  - a. First Name (firstName)
  - b. Last Name (lastName)
  - c. Employee ID (id)
  - d. Start Year (startYear)
  - e. Starting Salary (startSalary)
  - f. Current Salary (currentSalary)
  - g. next
- 3) Create a library of functions that operate on this data structure. The source code for the functions should be in lastname\_employeeec.c and the function prototypes should be included in lastname\_employeeec.h. The following functions should be in the library:
  - a. sEMPLOYEE \*create\_employee\_record() – allocates memory for a new employee record, prompts user, through the console, to enter the data for the employee, returns a pointer to the newly created employee record
  - b. sEMPLOYEE \*add\_employee\_record(sEMPLOYEE \*employeeListHead, sEMPLOYEE \*employee) – adds the employee record to a linked list; returns the new list head (it might have changed)
  - c. sEMPLOYEE \*delete\_employee\_record(sEMPLOYEE \*employeeListHead, unsigned int id) – deletes the employee record with the specified employee ID (can linear search to find record); returns the new list head (it might have changed)
  - d. void print\_employee\_record(sEMPLOYEE \*employee) – prints the data in the employee record
  - e. sEMPLOYEE \*sort\_employee\_records(sEMPLOYEE \*employeeListHead) – sorts the list of employee records according to last name (can use bubble sort); returns the new list head (it might have changed)
  - f. int write\_employee\_records(char \*filename, sEMPLOYEE \*employeeListHead) – writes the list of employee records to a file; returns 0 on SUCCESS, 1 on FAILURE

- g. sEMPLOYEE \*read\_employee\_records(char \*filename, sEMPLOYEE \*employeeListHead) – reads employee records from a file and adds them to the specified linked list; the new list head (it might have changed)
- 4) Create an application with source code in lastname\_main.c that is an employee record management system like the student record management system we developed in class. The application should give users a menu of choices:
  - 1 – add a new employee record
  - 2 – delete an employee record
  - 3 – sort employee records by last name
  - 4 – print all employee records
  - 5 – write all employee records to a file
  - 6 – read employee records from a file
  - 7 – exit
- 5) Create a Makefile to build two versions of your application: a normal version (employee) and a debug version (employee\_dbg) that is ready for use in GDB. Your Makefile should be designed so that a source file is compiled only if it has been modified.

**Deliverables:**

- 1) Submit a zip file named lastname\_assignment6.zip that contains: Makefile, lastname\_employeeec.h, lastname\_employeeec.c, lastname\_main.c. Be sure all of the source files are submitted **with a good set of comments to explain everything**. Submit this zip file named through Blackboard.

**Scoring:**

Your grade for this assignment will be determined by three criteria. The first criterion determines if your program compiles and runs producing the correct result. The correct result must adhere to what is specified in the **Tasks** section. The second criterion is whether the program follows the interface specification outlined in the **Tasks** section. The third criterion determines if your source code is well documented. Your source code must include (at the top) your name, class section, due date, assigned date, and a small description of your program. For this assignment, each line of code should have a descriptive comment.

Operation/Successful Demonstration	100%
Makefile works as required? 10%	
Render menu and basic operation? 10%	
Successfully add employee records to linked list and print employee records? 30%	

Successfully read/write employee records to/from a file? 20%	
Successfully find and delete an employee record by employee ID? 10%	
Successfully sort employee records by last name? 20%	
<b>Comments</b>	<b>10%</b>
Is the source code well-documented? 10%	
<b>Lateness</b>	<b>10% per day (including weekends and holidays)</b>