# HOMEWORK 8 DYNAMIC LINKED LISTS WITH POINTERS

Write a C program that will calculate the gross pay of a set of employees. Note that there are two items that will help you with this assignment.

1) Please use the template provide in the Homework 8 folder. Use this template "as is" to jump start your success in completing this homework.  It should compile and run right out the box.  
  
2) Watch the videos this week to help you better visualize and understand how the template code works.  Also, read through and make sure you understand the concepts in the lecture notes.

## ****WHAT YOU NEED TO DO:****

The program should prompt the user to enter the number of hours each employee worked. The output should be similar to your previous homework.

The program determines the overtime hours (anything over 40 hours), the gross pay and then outputs a table in the following format. Column alignment, leading zeros in Clock#, and zero suppression in float fields is important. Use 1.5 as the overtime pay factor.  
  
DO continue to use features such as functions and constants, and continue to follow our coding standards.

You should implement this program using the following structure to store the information for each employee.

struct employee  
{  
    char first\_name [10];  
    char last\_name [10];  
    int id\_number;             /\* use can use long int if you wish \*/  
    float wage;  
    float hours;  
    float overtime;  
    float gross;  
  
**struct employee \*next;**  
};

Create a linked list of structures using the following data:  
  
Connie Cobol 98401 10.60  
Mary Apl 526488 9.75  
Frank Fortran 765349 10.50  
Jeff Ada 34645 12.25  
Anton Pascal 127615 8.35  
  
Unlike previous homework, you need to prompt the user for all of the above information, ... and you still need to prompt for the hours worked for each employee.

**Hint:**Use one or two **scanf** statements to read in the first and last names with the %s format.

Get the data above from the terminal, and for each one:

* get dynamic memory, using **malloc**, for an employee node
* put the employee data in the dynamic memory node
* link the nodes with pointers in the above order

After the list pointers are in place you must get at the second and later instances of the structure by going from the first structure down the chain of list pointers.

Then, for each employee, read in the hours worked from the terminal. Do all appropriate computations, and write out the table.  
  
You do not need an array of structures like you used in homework 6 and 7. Use the template and dynamically allocate linked list nodes as needed.  Similar to the previous homework assignment:  
  
    a) Add a **Total** row at the end to sum up the hours, overtime, and gross columns  
    b) Add an **Average** row to print out the average of the hours, overtime, and gross columns.

Your code should work for any number of employees, and that is how the template is designed.  
  
**Tip:**Use left justification to line up character array name values ... for example: %-10.10s or %-10s

**Remember:**Use the Template!

# Optional Challenges

**1) Basic Optional Challenges (Bronze Level):**

    a) Calculate and print the **minimum** wage, hours, overtime, and gross values  
    b) Calculate and print the **maximum** wage hours, overtime, and gross values **2) Intermediate Optional Challenge (Silver Level):**

Instead of building the linked list in the main function, call another function from main that prompts for the various input (name, clock, wage) and then returns back the pointer.

**3) Advanced Optional Challenge (Gold Level):**

Read the Input/Output chapter in the Kochan book (Ch 15 in the 4th edition, or Ch 16 in the third edition). See if you can read the information from a text file instead of the screen. You can skip reading data from the screen if you decide to do the challenge. You will need to use file pointers here for opening up a file on your computer to read the data and optionally, an output file on your computer to print a report of your employee data.  Remember that IDEOne will not support reading files from your computer, but it should work well with any compiler you have installed locally on your computer.

You can also try to incorporate any of the previous challenges from the past homeworks as well, and it would be an alternative if you are using IDEOne.

## Additional Optional Challenge

If you want an additional challenge, a code template has also been provided that starts you in the right direction if you want to implement the homework using separate C source files for each function as well as including a header file as needed. You can load the multiple template files provided into your native compiler and Integrated Development Environment (IDE) to get started.